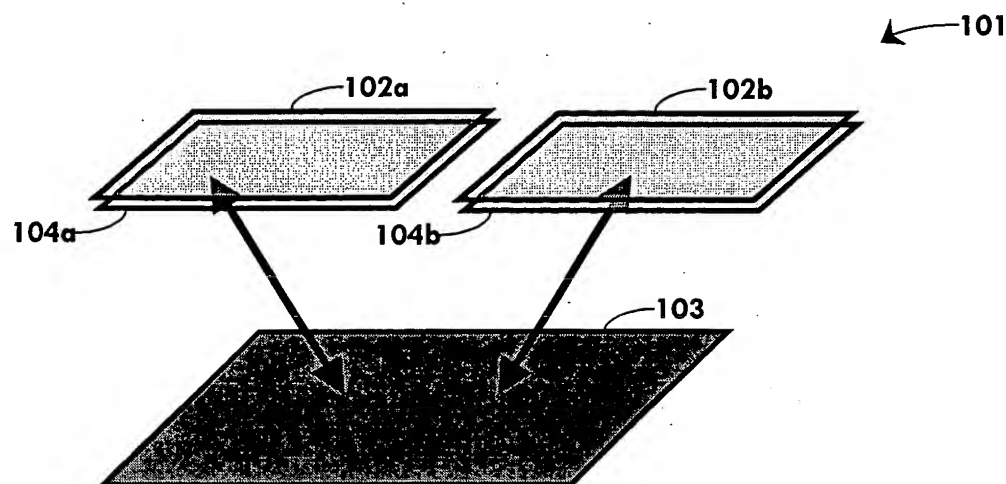# Figures

Figure 1

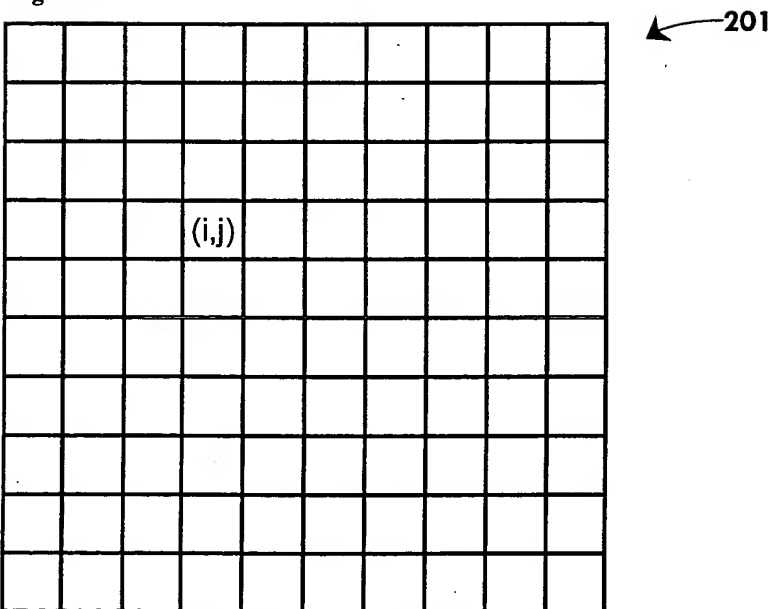Figure 2

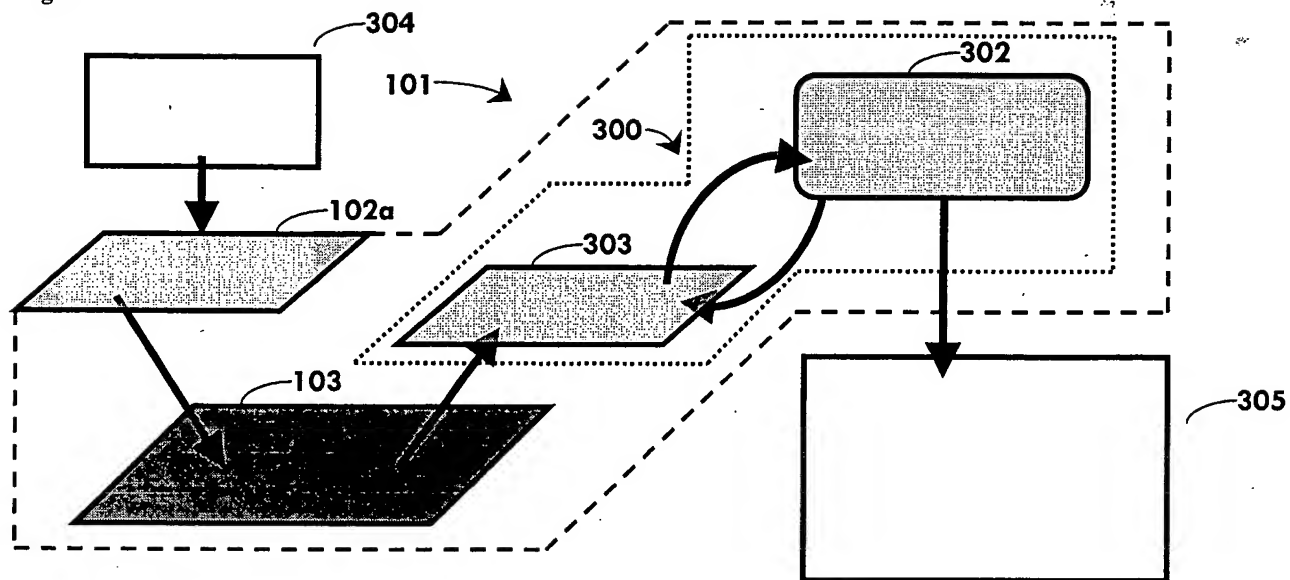Figure 3

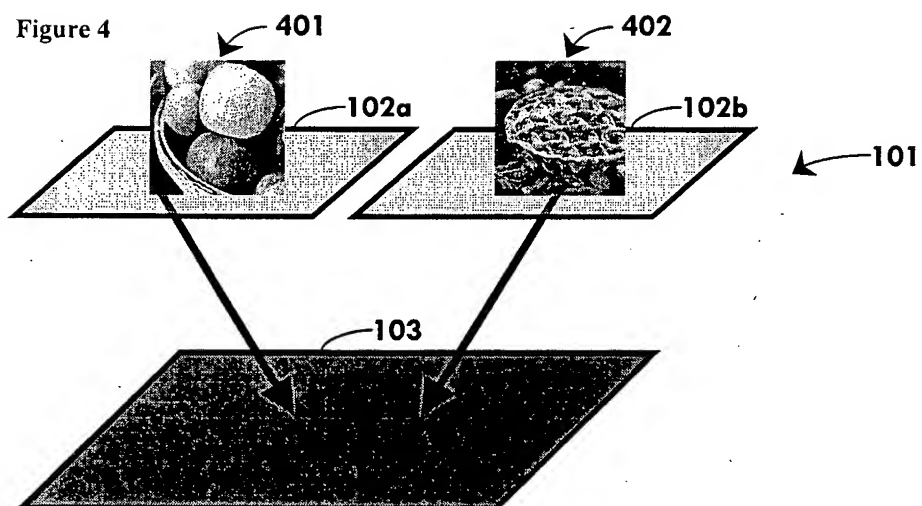
Figure 4

Figure 5

401

402

102a

104b

101

103

Figure 6

401

402

104a

102b

101

103

Figure 7

101

401

704

"apples" — 703

102a

01100011 — 702

301

103

Figure 8

101   "apples" ; 95% certainty — 801

401

704

102a

01100011 — 702

303

103
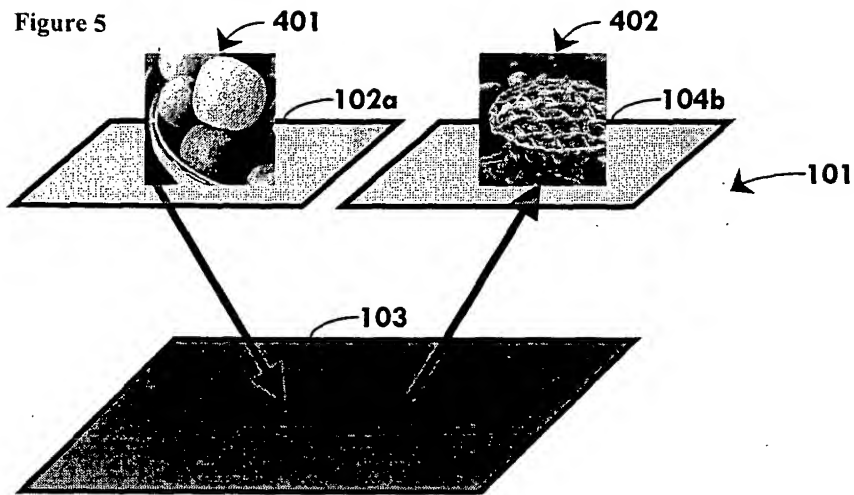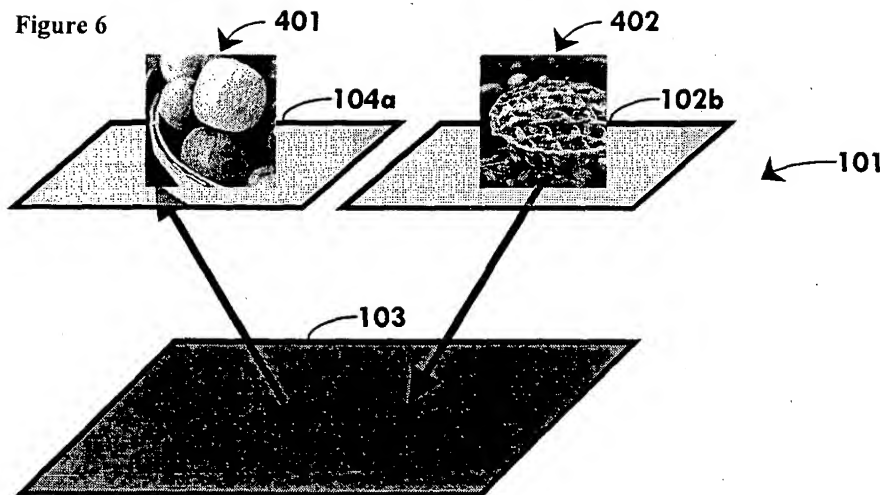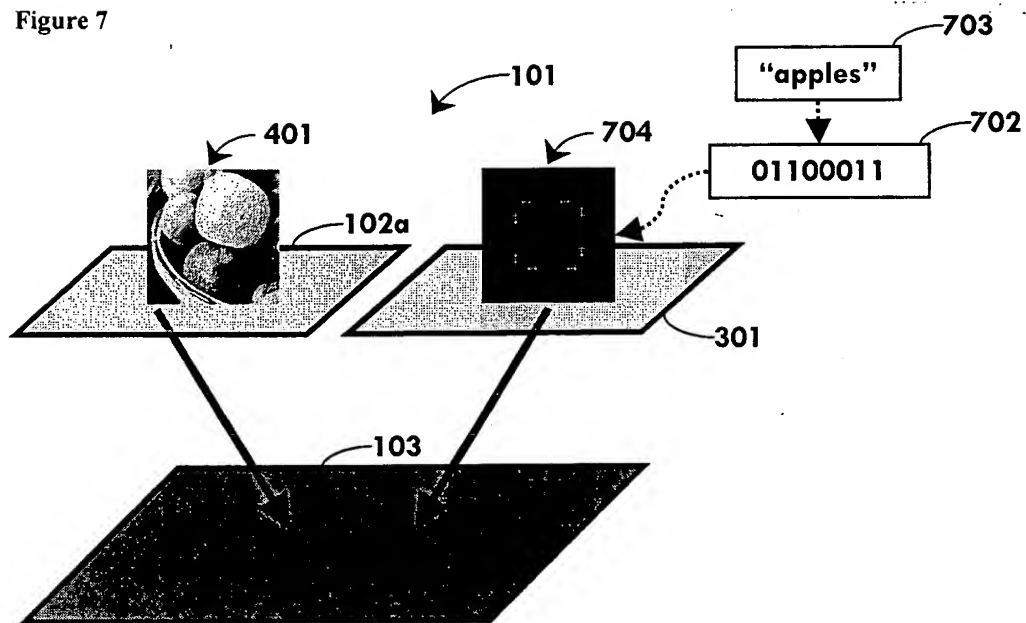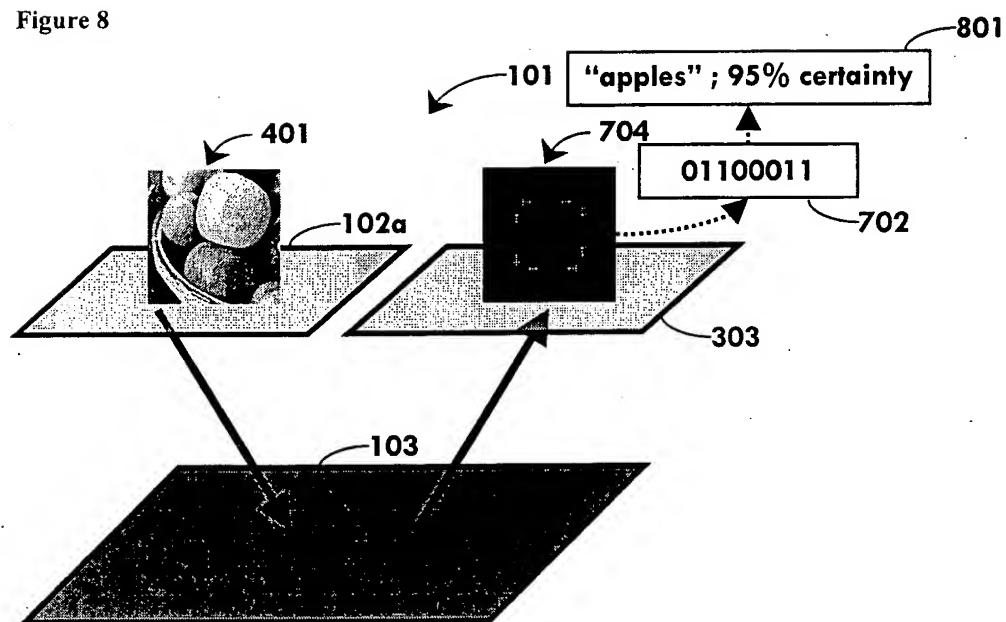
Figure 9

Input information to a buffer

Has a new input (P$_i$) been detected? — 901

No

Yes

Load pattern P$_i$ into Buffer 102 — 902

Pattern array P$_i$ on Buffer 102 — 903

(Optional) Perform preprocessing on pattern P$_i$ — 904

Is P$_i$ complex? — 905

No

Assign phase to P$_i$ to make it a complex array — 906

Figure 10

Transport information from buffer to cortex or cortex to buffer

┌─────────────────────────────┐ ⎯1001
│   Propagate P_i from Buffer 102   │
│   to Cortex 103   │
└─────────────────────────────┘

⎯1002
Pattern array P_i on Cortex 103

⎯1003
(Optional) Store the
propagated pattern

Figure 11
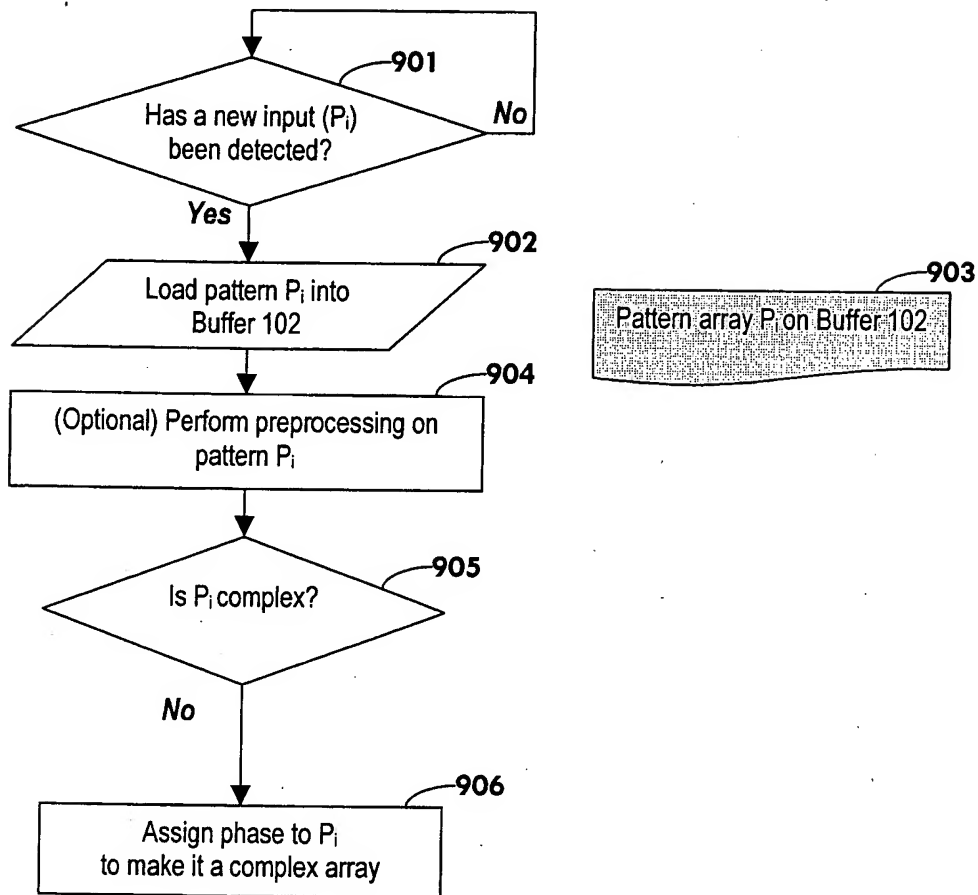
⎯102a

⎯103

Figure 12

⎯102a

⎯103
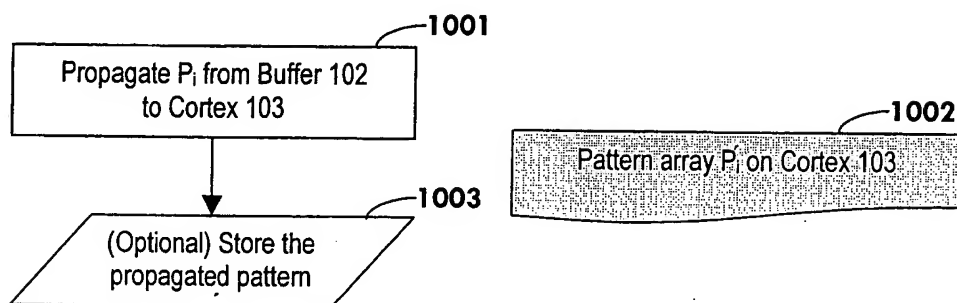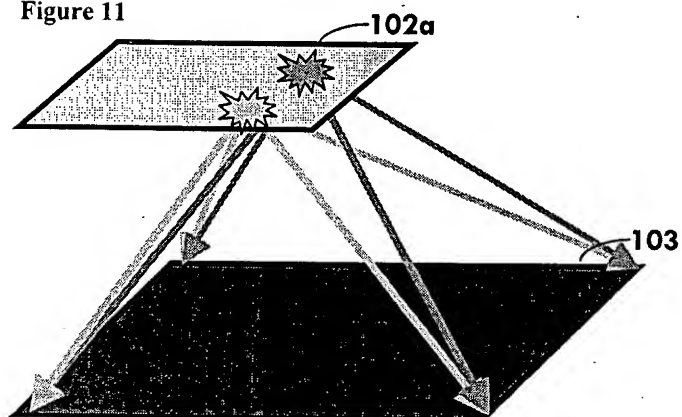
Figure 13

Associate two or mor patt rns

```
                    ┌─────────────────────────┐  ╱⌐1301
                   ╱   Load a propagated      ╱
                  ╱    pattern P'ᵢ into memory, ╱
                 ╱     if necessary           ╱
                └─────────────────────────┘
                            │
                            ▼
                         ╱╲                              ╱⌐1302
                       ╱    ╲ ⌐1303          ┌─────────────────────────────┐
                     ╱        ╲              │  Pattern array Pᵢ' on Cortex 103 │
                   ╱  Have all   ╲    No      └─────────────────────────────┘
                  ╱  patterns to be ╲ ─────────┐
                   ╲  associated   ╱          │
                     ╲  loaded?  ╱            │
                       ╲    ╱                  │
                         ╲╱                    │
                          │ Yes                │
                          ▼                    │
                                               │
                    ┌──────────────────┐  ╱⌐1304
                    │ Associate the selected patterns │
                    │ together via an invertible      │
                    │ mathematical operation          │
                    └──────────────────┘
                            │
                            ▼
                                              ┌─────────────────────────────┐ ⌐1305
                    ┌──────────────────┐  ╱⌐1306    New association          │
                   ╱  (Optional) Store the ╱    ⟨P'ᵢ, P'ⱼ, ..., Pᵢ⟩ on Cortex 103 │
                  ╱   new association(s)  ╱    └─────────────────────────────┘
                 └──────────────────┘
```
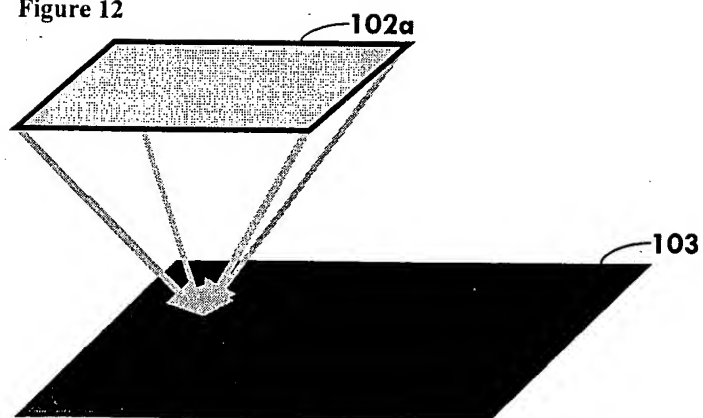
Load a propagated pattern P'ᵢ into memory, if necessary — 1301

Have all patterns to be associated been loaded? — 1303

No

Pattern array Pᵢ' on Cortex 103 — 1302

Yes

Associate the selected patterns together via an invertible mathematical operation — 1304

New association ⟨P'ᵢ, P'ⱼ, ..., Pᵢ⟩ on Cortex 103 — 1305

(Optional) Store the new association(s) — 1306

Figure 14

P_i

102a

P_i

102b

103

$\langle P'_i, P'_i \rangle$

Figure 15

Superpose two or mor associations

Load an association into memory, if necessary ⌐1501

⌐1502

mth association
$\langle P^i_{j,m}, P^{[j}_{j,m}, ..., P^{[j}_{j,m} \rangle$ on Cortex
103

Have all appropriate associations been loaded? ⌐1503

No

Yes

Form a linear combination of the appropriate associations ⌐1504

⌐1505

Updated Cortex 103

(Optional) Store the updated Cortex 103 ⌐1506
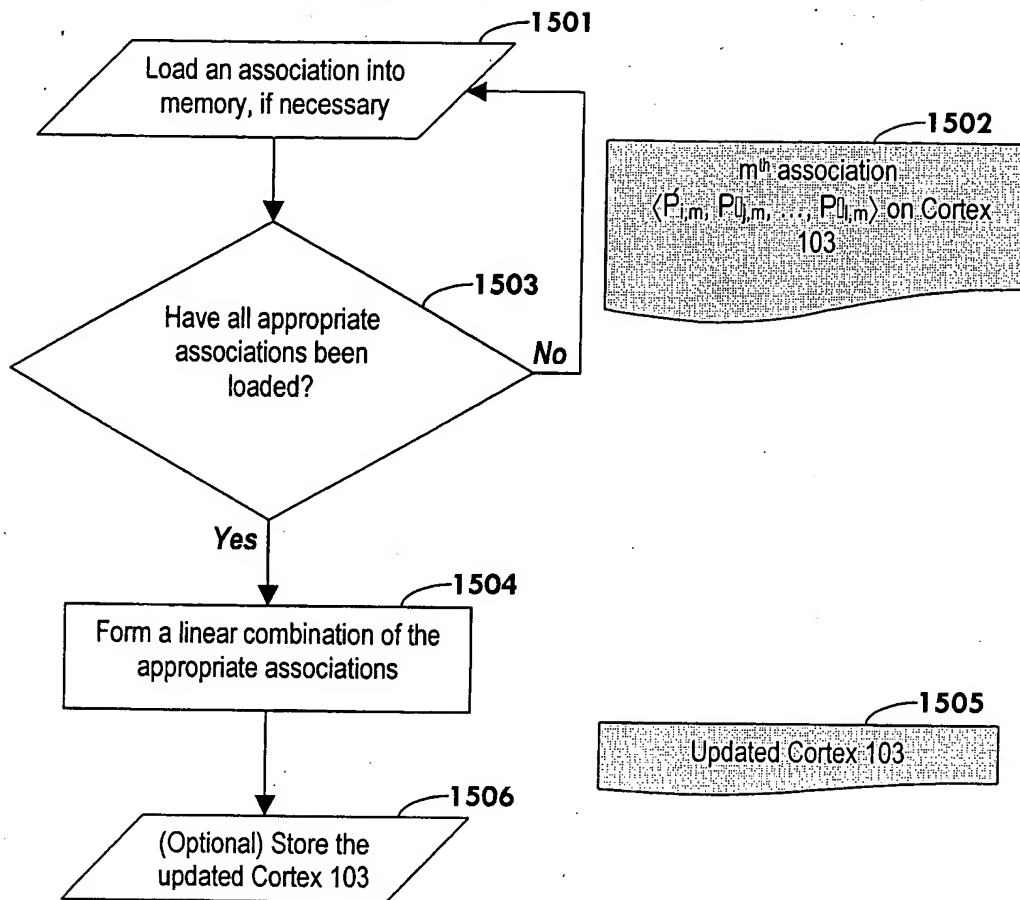
Figure 16

De-associate a pattern from a stored cortex

Load a prompting pattern $\dot{R}_{i,n}$ into memory, if necessary — 1601

Have all patterns for de-association been loaded? — 1603

No

Prompting pattern array $R_{i,n}$ on Cortex 103 — 1602

Yes

Perform de-association using the inverse of the association function — 1604

De-associated wave D on Cortex 103 — 1605

(Optional) Store the de-associated wave $\acute{D}$ — 1606

Figure 17

$R_{i,n}$

102a

103

$$\acute{D} = \langle C \mid \acute{R}_{i,n} \rangle$$

Figure 18

Export buffer information

Retrieved wave D evaluated on Buffer 104b
1801

Convert values of retrieved wave D to original form
1802

Converted form of D
1803

(Optional) Store the converted wave as data file
1804

Figure 19

Generate an internal identification pattern (IIP)

```
                                          ⌐1901
   ┌─────────────────────────────┐
   │ Select a unique, machine-readable │
   │      identification code          │
   └─────────────────────────────┘
                  │
                  │              ⌐1903
                  ▼
   ┌─────────────────────────────┐
   │ Generate a unique, machine- │
   │ readable internal identification │
   │       pattern (IIP)         │
   └─────────────────────────────┘
                  │
                  │       ⌐1905
                  ▼
      /─────────────────────/
     / (Optional) Store the  /
    /   identification code  /
   /        and IIP         /
  /───────────────────────/
```

⌐1902

Identification code Lᵢ

⌐1904

Internal identifier pattern (IIP) Iᵢ

Figure 20

Figure 21

Decode a r trieved int rnal identification pattern (IIP)

2101
Read in a retrieved IIP, J_j

2102
Retrieved IIP, J_j

2103
Analyze the retrieved IIP

2104
Extract redundant candidate identification codes from the IIP

2105
Candidate identification codes $\{L_1, ..., L_n\}$

2106
Calculate confidence estimate

2107
Is confidence estimate high enough to make an identification?

2108
Is the extracted code valid?

No

Yes

No

Yes

2109
Report identification code L_k and a confidence estimate E

2110
Has the maximum number of attempts been made?

Yes

No

2111
Report failure to identify a pattern associated with the retrieved IIP .J_j

2112
Re-analyze the IIP to yield an alternative set of identification codes

2113
Alternative candidate identification codes $\{L_1, ..., L_n\}$

2114
Is the new set of codes different from the last set?

No

Yes

Figure 22

Learn

Load an input pattern $P_{i,n}$ into memory to be learned according to Figure 9 ⟋—2201

Input pattern array $P_{i,n}$ on Buffer 102 ⟋—2202

Have all patterns to be associated with $P_{i,n}$ been loaded? ⟋—2203 — **No**

Yes

Propagate patterns $P_{i,n}$, $P_{j,n}$,..., $P_{l,n}$ to Cortex 103 according to Figure 10 ⟋—2204

Propagated, input patterns $P_{i,n}$, $P_{j,n}$, ..., $P_{l,n}$ on Cortex 103 ⟋—2205

Perform association of $P'_{i,n}$, $P'_{j,n}$, ...,$P'_{l,n}$ according to Figure 13 ⟋—2206

New association $\langle P'_{i,n}, P'_{j,n}, ..., P'_{l,n} \rangle$ on Cortex 103 ⟋—2207

(Optional) Superpose the new association with previously stored associations according to Figure 15 ⟋—2208

Updated Cortex 103 ⟋—2209

Yes

Do any patterns remain to be learned? ⟋—2210

No

(Optional) Store the updated Cortex 103 ⟋—2211

Figure 23

Recognize

2301
Load a prompting pattern $R_{i,n}$ into memory, if necessary, to make a retrieval, according to Figure 9

2302
Prompting pattern array $R_{i,n}$ on Buffer 102

2303
Transport $R_{i,n}$ to Cortex 103 according to Figure 10

2304
Propagated, prompting pattern array $R_{i,n}$ on Cortex 103

2305
Perform de-association of Cortex 103 using $R_{i,n}$ according to Figure 16

2306
De-associated wave D on Cortex 103

2307
Propagate complex conjugate of de-associated wave D to the appropriate buffer(s)

2308
Retrieved wave D evaluated on one or more buffers

2309
Convert retrieved wave D to original field form according to Figure 18

2310
Retrieved field D

2311
(Optional) Store the retrieval

Figure 24

Learn for autonomous id ntification 〜2401

Load an input pattern $P_{i,n}$ into memory to be learned, according to Figure 9

〜2403

Have all patterns to be associated with $P_{i,n}$ been loaded? **No**

Yes 〜2404

Generate an IIP for $P_{i,n}$, $P_{j,n}$,...,$P_{l,n}$ according to

〜2406

Gather registration information

〜2408

Transport patterns $P_{i,n}$, $P_{j,n}$,...,$P_{l,n}$ and IIP I_n to Cortex 103 according to Figure 10

〜2410

Perform association of $P_{i,n}'$, $P_{j,n}'$, ..., $P_{l,n}'$ and $\hat{I}_n$ according to Figure 13

〜2412

(Optional) Superpose the new association with previously stored associations according to Figure 15

Yes 〜2414

Do any patterns remain to be learned?

No 〜2415

(Optional) Store the updated Cortex 103

〜2402

Input pattern array $P_{i,n}$ on Buffer 102

〜2405

IIP I_n on Buffer 301

〜2407

Associated files and commands

〜2409

Propagated, input patterns $P'_{i,n}$, $P'_{j,n}$, ...,$P'_{l,n}$ and IIP I_n on Cortex 103

〜2411

New association $\langle P_{i,n}', P_{j,n}', ..., P_{l,n}', I_n \rangle$ on Cortex 103

〜2413

Updated Cortex 103

Figure 25

Identify and respond autonomously

2501 Load a pattern $R_{i,n}$ into memory to be identified, according to Figure 9

2502 Prompting pattern array $R_{i,n}$ on Buffer 102

2503 Recognize $R_{i,n}$ according to Figure 23.

2504 Retrieved IIP, J_n

2505 Decode IIP J_n according to Figure 21.

2506 Reported identification code L_n and confidence estimate E_n

2507 Retrieve files based on the identification and the confidence estimate.

2508 Files associated with identification code L_n

2509 (Optional) Execute or output the commands contained in the retrieved files.

2510 Associated commands

2511 Do any patterns remain to be identified?

Yes

No

2512 (Optional) Store the identification log

2513 Log files

Figure 26

Adaptively improv identification performance through xperi nce

Identify pattern $R_{i,n}$ according to Figure 25.

2601

2603

Monitor correctness of system's identification

2602

Reported identification code L_n and confidence estimate E_n

2604

Yes

Is the identification correct?

No

2605

Type of error made?

2606

Incorrect ID, IIP I_p returned ($p \neq m$)

False positive

False negative

2611

False rejection or correct ID but confidence estimate too low

1

2

Figure 26 (cont.)

Adaptively improve identification performance through experi nc , continued

2606

Incorrect ID, IIP I_p returned (p≠m)

False positive

1

2607

Adjust cortex to weaken association of $R_{i,n}$ with $P_{i,m}$

2608

Re-introduce pattern $R_{i,n}$ to be identified

No

2609

Is the identification correct?

Yes

2610

Store updated Cortex 103

Figure 26 (cont)

Adaptively improve identification performance through experience, continued

2611

False rejection
or
correct ID but
confidence estimate
too low

2

False negative

2612

Incorporate   Mode to adjust
Cortex 103?   *Replace*

2613

Broaden the association of
$P_{i,m}$ with pattern $P_{i,q}$

2614

Replace the original pattern
$P_{i,m}$ with $P_{i,q}$

Re-introduce pattern $R_{i,n}$ to
be identified

2615

2616

Is the identification strong
enough?

No

Yes

2610

Store updated
Cortex 103

Figure 27

Associative memory system block diagram

Identification System Controller — 2705

System Memory — 2706

Processor(s) — 2707

Storage Device(s) — 2708

Output Device(s) — 2709

Input Device — 2701a

Buffer — 2702a

2703

Input Device — 2701b

Buffer — 2702b

Cortex — 103

System Bus — 2710

IIP Buffer — 2704

IIP Generator Module — 2721

IIP Reader Module — 2722

Autonomous Execution Module — 2723

Adaptive Learning Module — 2724

Utilizing Apparatus or System(s) — 2725

300

2700

Figure 28

Buffer block diagram for physical wave-based embodiment

Input Device ⟋ 2701

Buffer ⟋ 2702 ⟵ 2800

Device Driver ⟋ 2803

A/D Converter ⟋ 2802

Preprocessor Module ⟋ 2804

Phase Assignment Module ⟋ 2805

Postprocessor Module ⟋ 2814

Identification System Controller ⟋ 2705

System Memory ⟋ 2706

System Bus ⟋ 2710

Processor(s) ⟋ 2707

Storage Device(s) ⟋ 2708

Main Buffer Memory ⟋ 2806

Receiver Array Controller ⟋ 2812

Broadcast Array Controller ⟋ 2813

Buffer Antenna Arrays ⟋ 2807

Receiving Antenna ⟋ 2808

Receiver Memory Element ⟋ 2809

Broadcast Antenna ⟋ 2810

Broadcast Memory Element ⟋ 2811

Figure 29

Cortex block diagram for physical wave-based embodiment

Identification System Controller — 2705

System Memory — 2706

Processor(s) — 2707

Storage Device(s) — 2708

System Bus — 2710

Association Module — 2909

De-association Module — 2911

Linear Combination Module — 2910

Complex Conjugation Module — 2915

Main Cortex Memory — 2901

Receiver Array Controller — 2907

Broadcast Array Controller — 2908

Receiving Antenna — 2903

Receiver Memory Element for Buffer n — 2904

Association Memory Element — 2913

Linear Combination Memory Element — 2914

Broadcast Antenna — 2905

Broadcast Memory Element — 2906

De-association Memory Element — 2912

Cortex Antenna Arrays — 2902

Cortex — 103

2900

Figure (30)

Buffer block diagram for software-based embodiment

Input Device — 2701

Device Driver — 2803

A/D Converter — 2802

Preprocessor Module — 2804

Phase Assignment Module — 2805

Postprocessor Module — 2814

Main Buffer Memory — 3001

Wave Propagation Calculation Module — 3002

Identification System Controller — 2705

System Memory — 2706

System Bus 2710

Processor(s) — 2707

Storage Device(s) — 2708

Receiver Memory Element — 2809

Broadcast Memory Element — 2811

Buffer Arrays — 3003

Buffer — 2702

3000

Cortex block diagram for software-based embodiment

Identification System Controller — 2705

System Memory — 2706

Processor(s) — 2707

Storage Device(s) — 2708

System Bus — 2710

Association Module — 2909

De-association Module — 2911

Linear Combination Module — 2910

Complex Conjugation Module — 2915

Main Cortex Memory — 3101

Wave Propagation Calculation Module — 3102

Receiver Memory Element for Buffer n — 2904

Broadcast Memory Element — 2906

Association Memory Element — 3102

De-association Memory Element — 2912

Linear Combination Memory Element — 2914

Cortex Arrays — 3103

Cortex — 103

3100

3100

Figure 32

Buffer block diagram for hardwired circuitry-based embodiment

Input Device — 2701

Device Driver — 2803

A/D Converter — 2802

Preprocessor Module — 2804

Phase Assignment Module — 2805

Postprocessor Module — 2814

Main Buffer Memory — 3201

Hardwired Wave Propagation Circuitry Module — 3202

Identification System Controller — 2705

System Memory — 2706

Processor(s) — 2707

Storage Device(s) — 2708

System Bus 2710

Receiver Memory Element — 2809

Broadcast Memory Element — 2811

Buffer Arrays — 3203

Buffer — 2702

3200

Figure (33)

Cortex block diagram for hardwired circuitry-based embodiment

Figure 34

Wave propagation circuitry block diagram for hardwired circuitry-based embodiment

Figure 35

3500

3501

Figure 36

3600

Figure 37

3700

3701
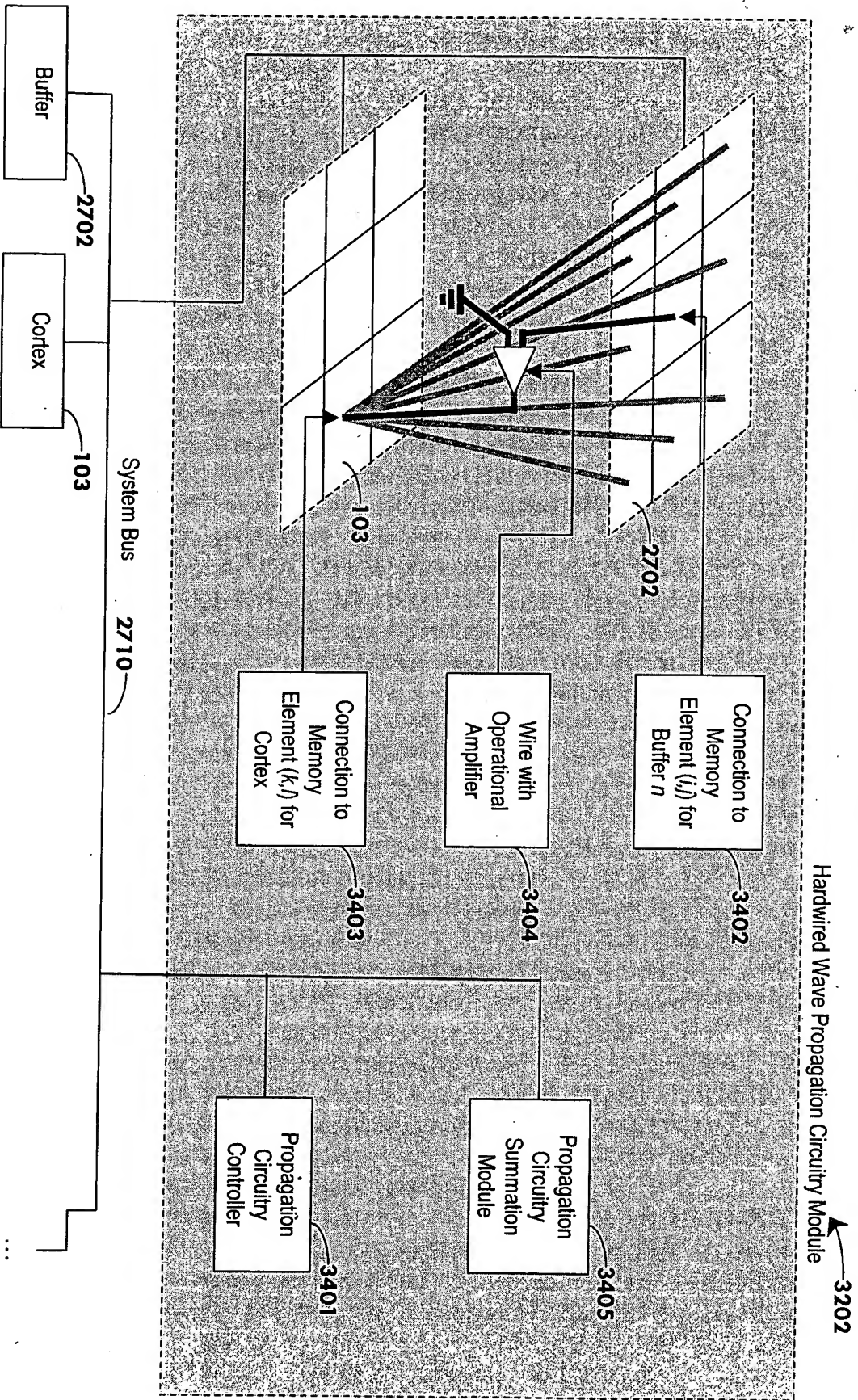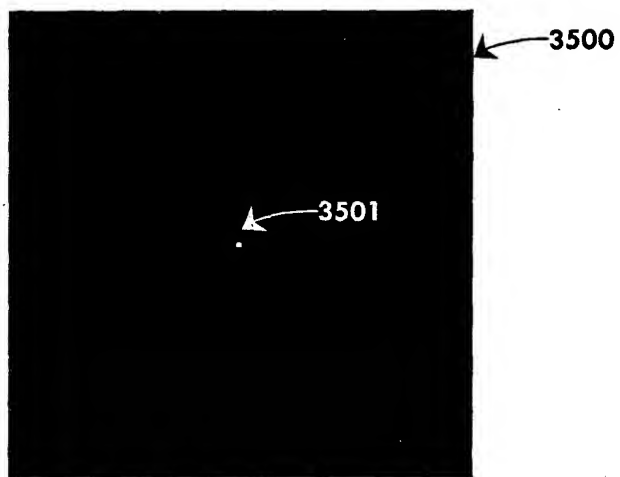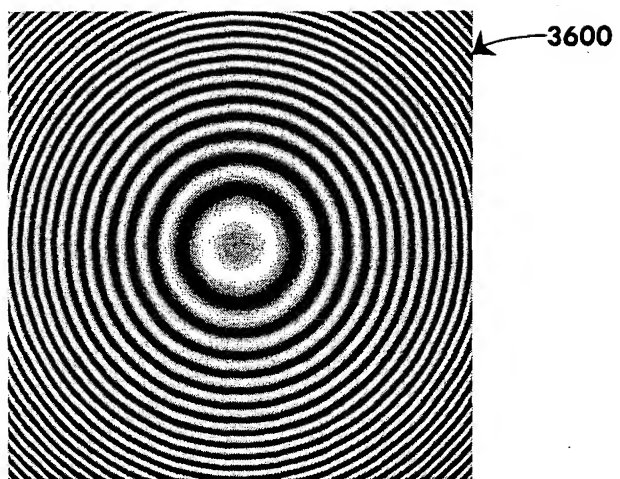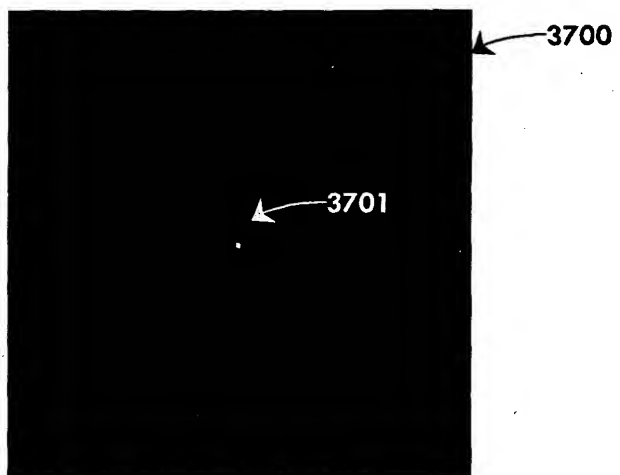
Figure 38

3800

Figure 39

3900

Steve
Kramer

Figure 40

4000

Figure 41
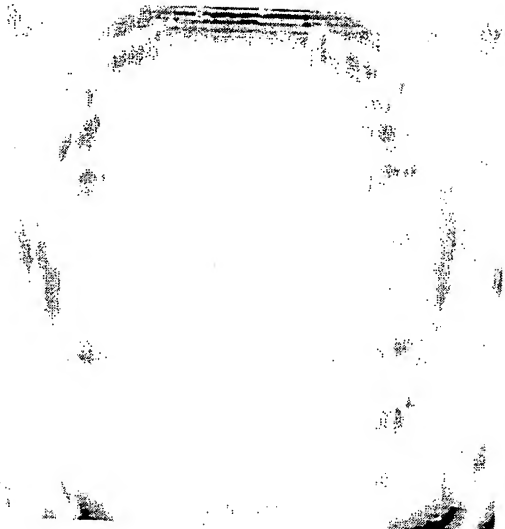
4100

Figure 42

4200

Steve Kramer
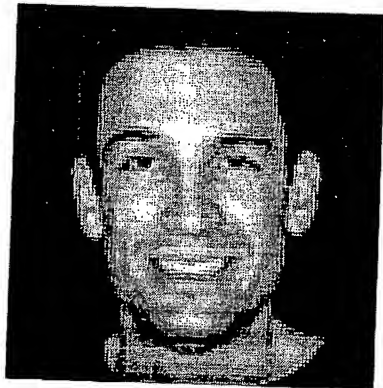
Figure 43

4300

Figure 44
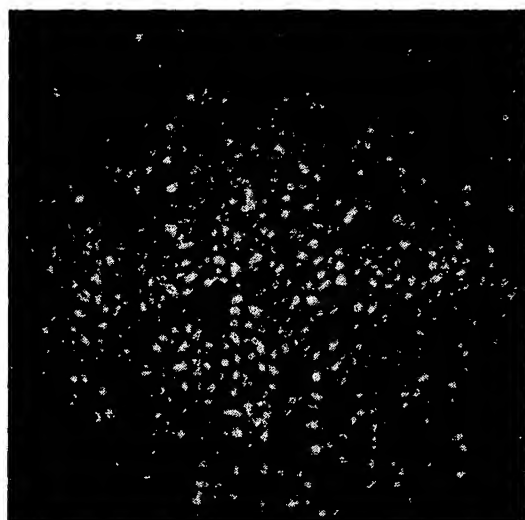
4400

Figure 45
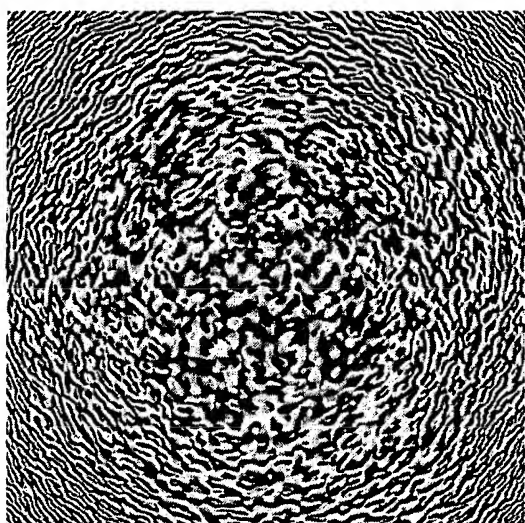
4500

Figure 46

4600

Figure 47

4700

Figure 48a

4800

Figure 48b
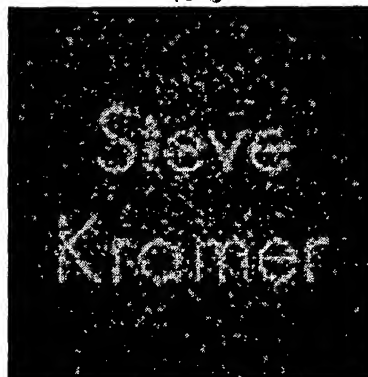
4801

Figure 49a

4900

Figure 49b
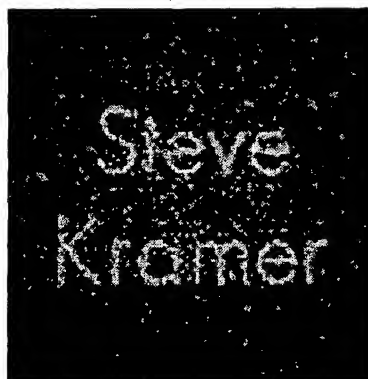
4901

Figure 50a
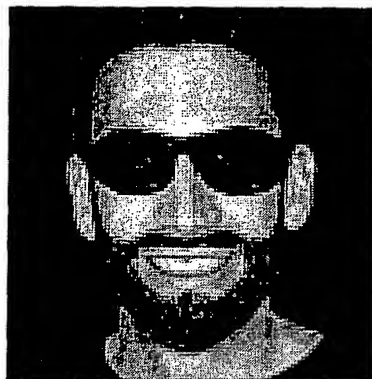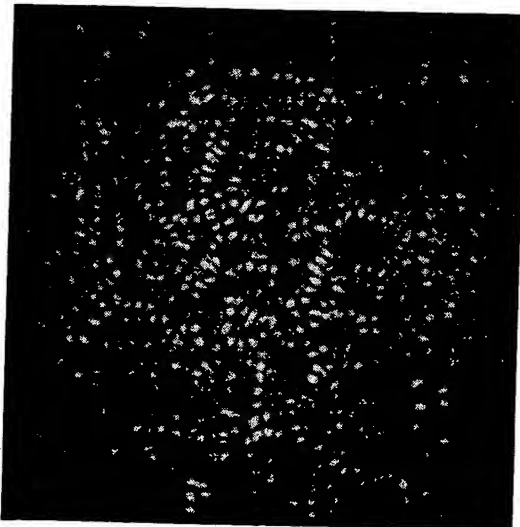

5000

Figure 50b
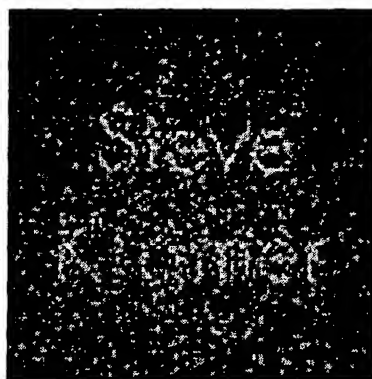
5001

Figure 51a

5100

Figure 51b
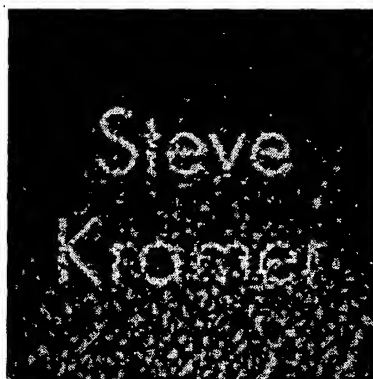
5101

Figure 52a

5200

Figure 52b
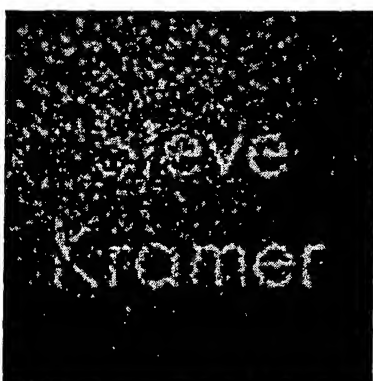
5201

Figure 53a

5300

Figure 53b
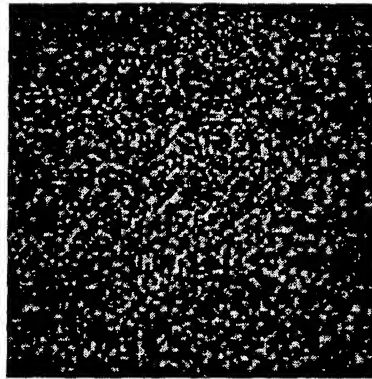
5301

Figure 54a

5400

Figure 54b
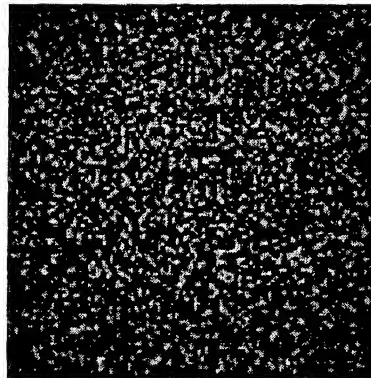
5401

Figure 55a

5500

Figure 55b
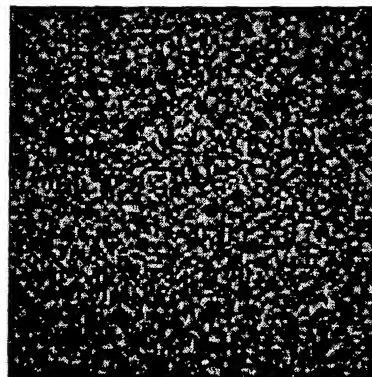
5501
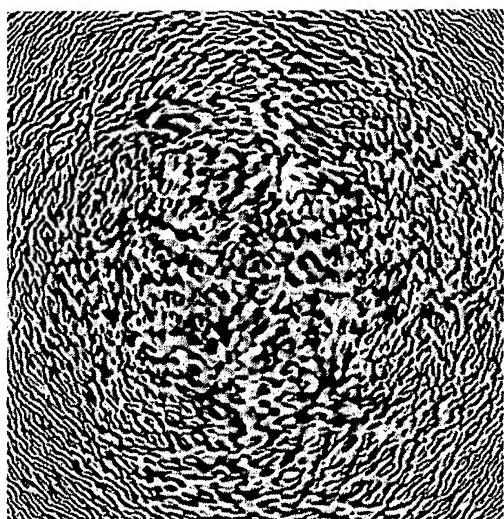
Figure **96**

5600
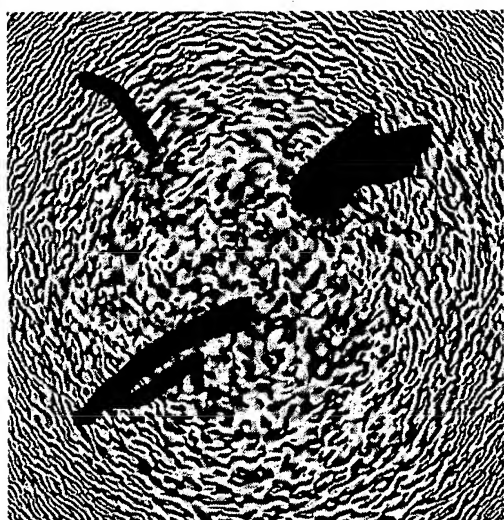
Figure **57**

5700

Figure 58a

3800

Figure 58b

5801

Figure 59a

4800

Figure 59b
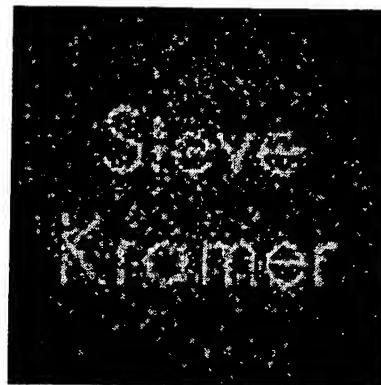
5901

Figure 60a
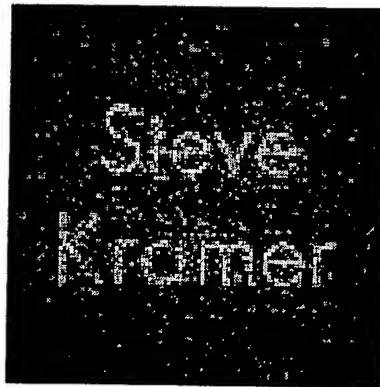
4900

Figure 60b

6001

Figure 61a
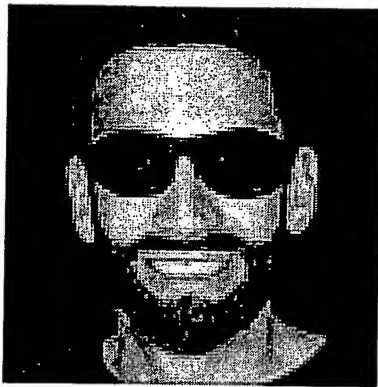
5000

Figure 61b
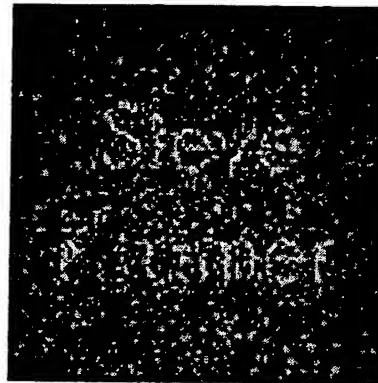
6101

Figure 62a
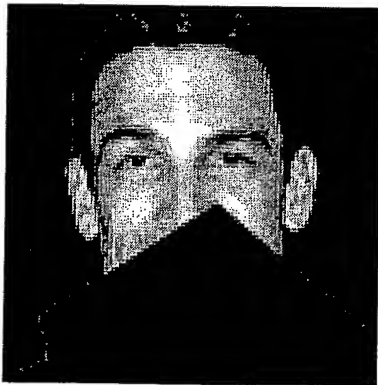
5100

Figure 62b
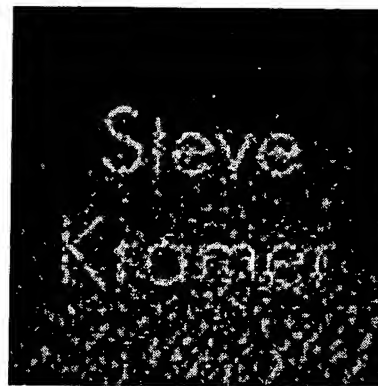
6201

Figure 63a

5200

Figure 63b
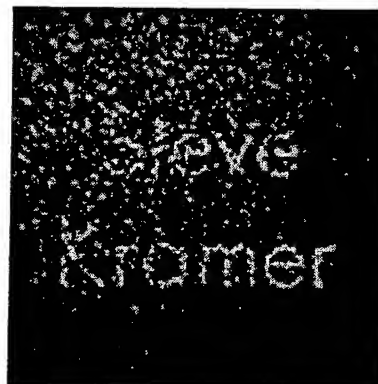
6301

Figure 64
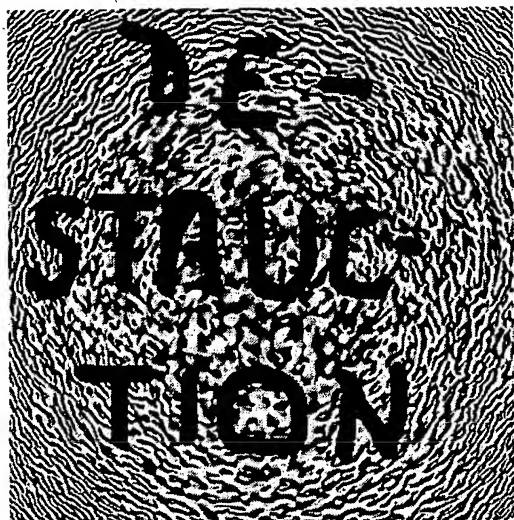
6400

Figure 65a

3800

Figure 65b
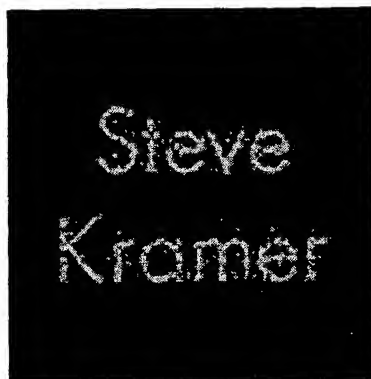
6501

Figure 66a

4800

Figure 66b
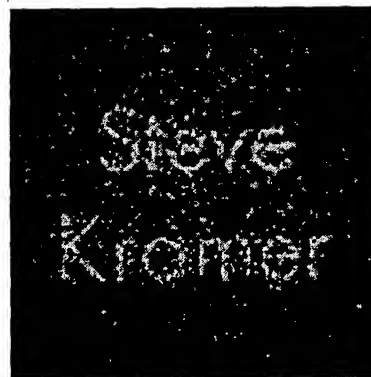
6601

Figure 67a 67a

←——4900

Figure 67b
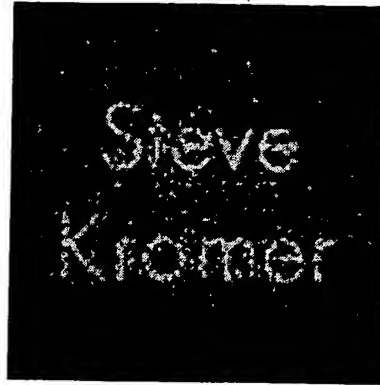
←——6701

Figure 68a 68a
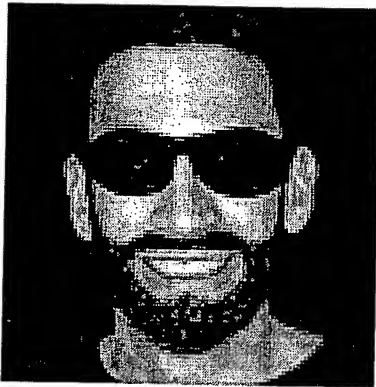
←——5000

Figure 68b
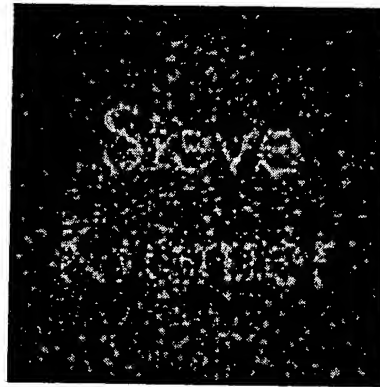
←——6801

Figure 69a 69a

←——5100

Figure 69b
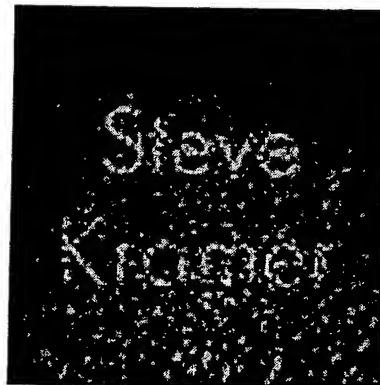
←——6901

Figure 70a

← 5200

Figure 70b
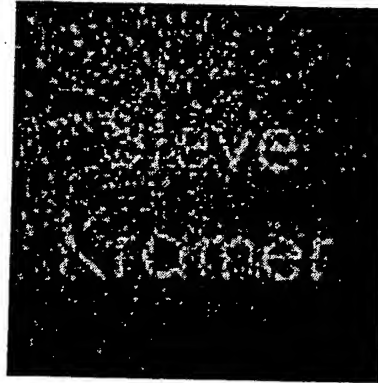
← 7001
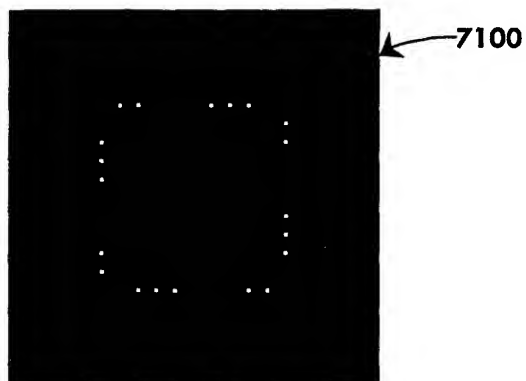
Figure 71
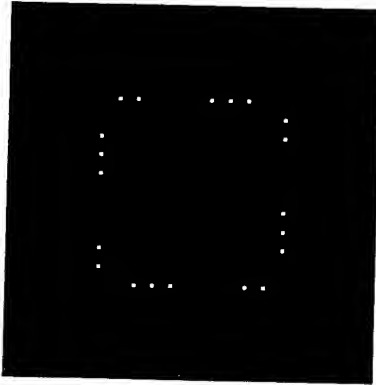
← 7100

Figure 72

← 7200

Figure 73
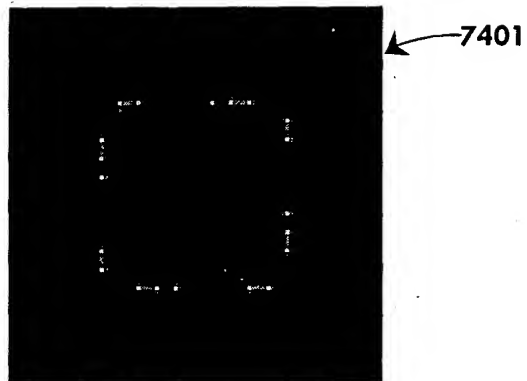

7300

Figure 74a

7100

Figure 74b
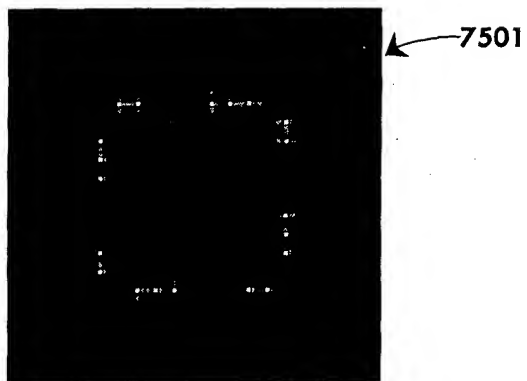

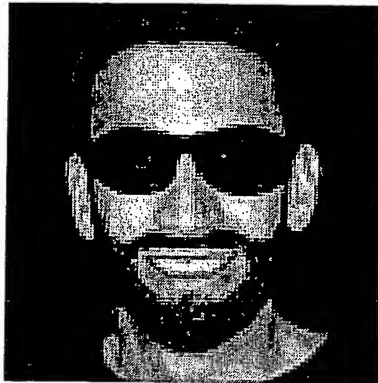7401

Figure 75a

4900

Figure 75b

7501
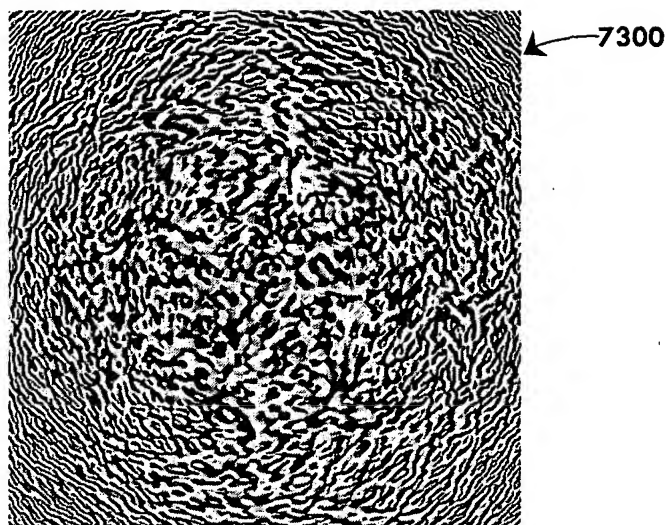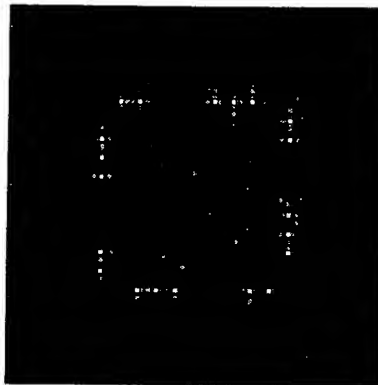
Figure 76a 769

←─5000

Figure 76b

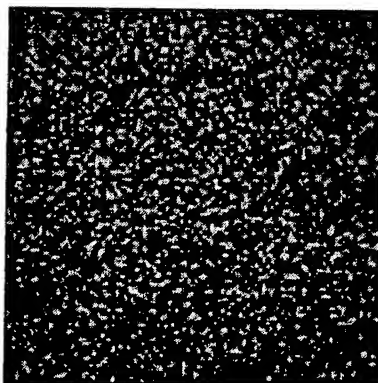←─7601

Figure 77a 77a

←─5200

Figure 77b

←─7701

Figure 78a 78a

←─5400

Figure 78b

←─7801

Figure 79

IIP Reconstruction (Steve K.)
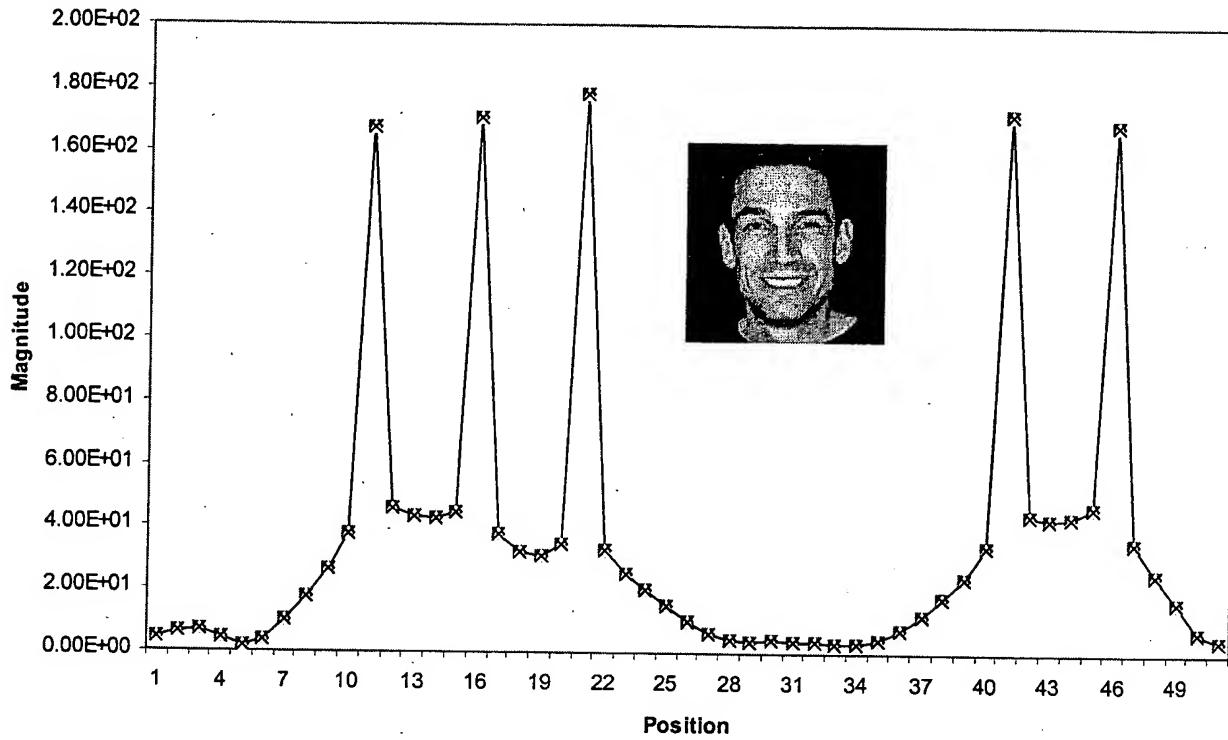
Figure 80

IIP Reconstructi n (Stev K 1)
4.1% obscured
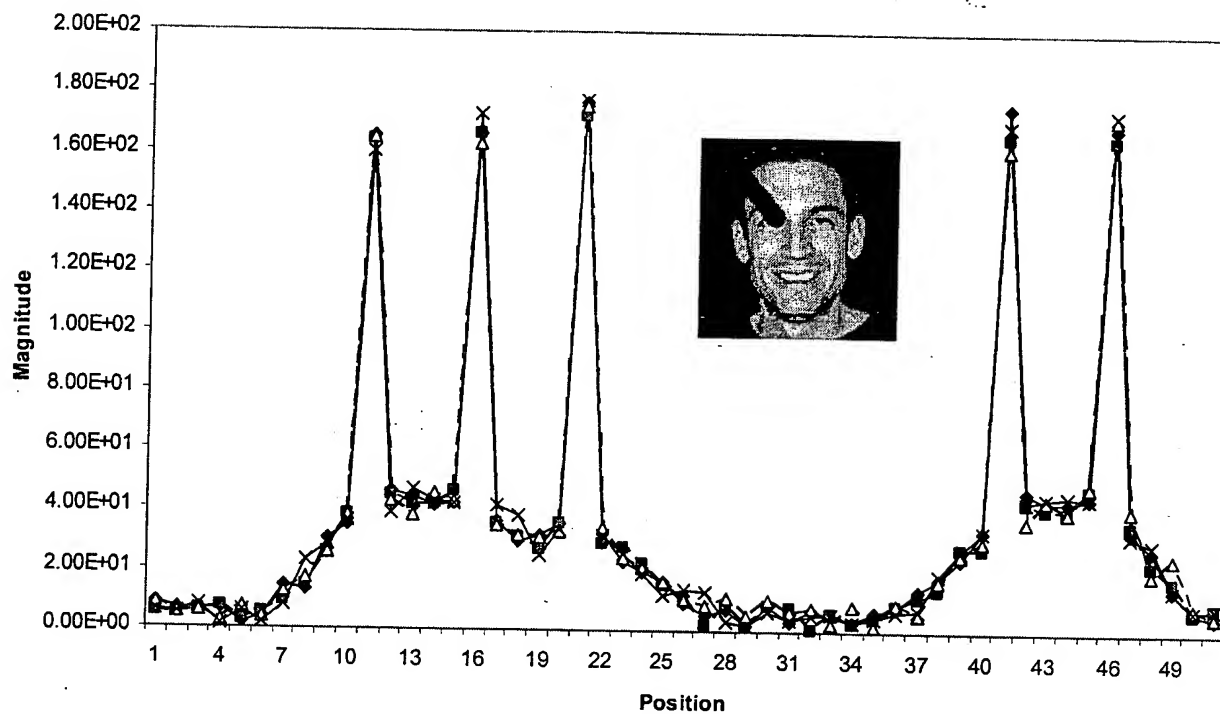
Figure 19

81

IIP Reconstruction (Stev K. 7)
33.4% bscured
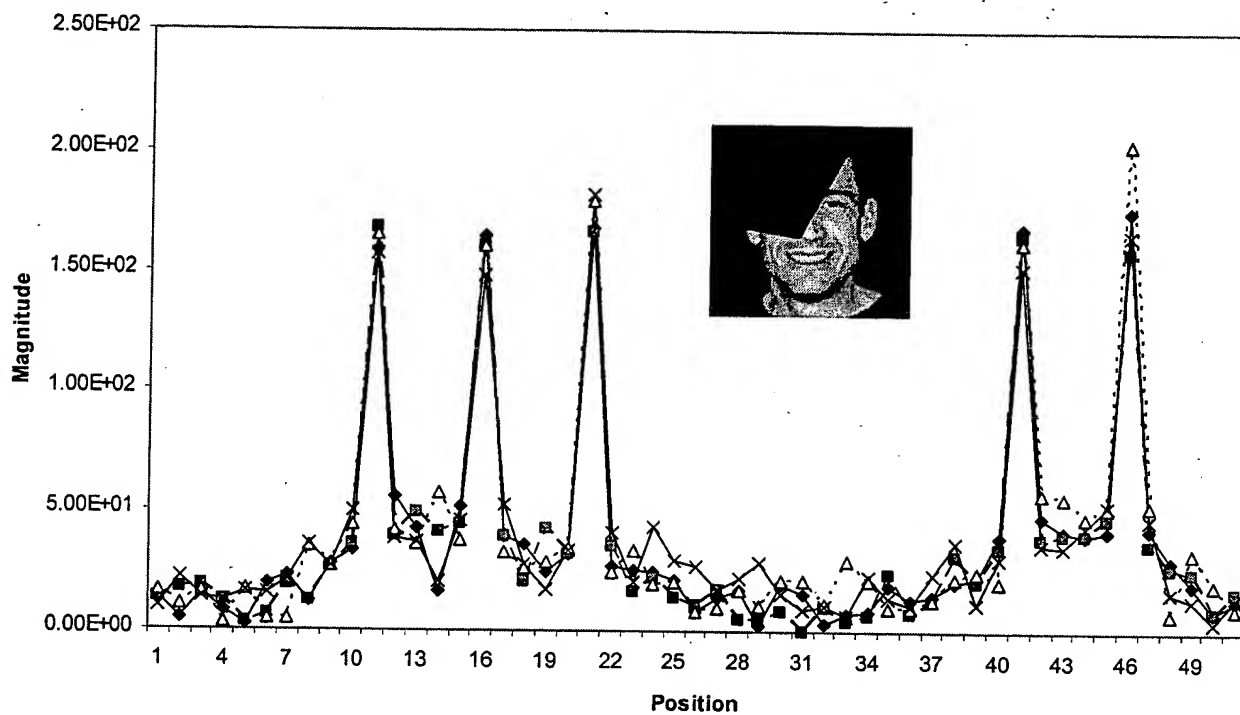
Figure

82

IIP Reconstruction as a Function of Degree of Obscurity

Percentage of Image Obscured

◄—3800

Figure 83b

◄—8301
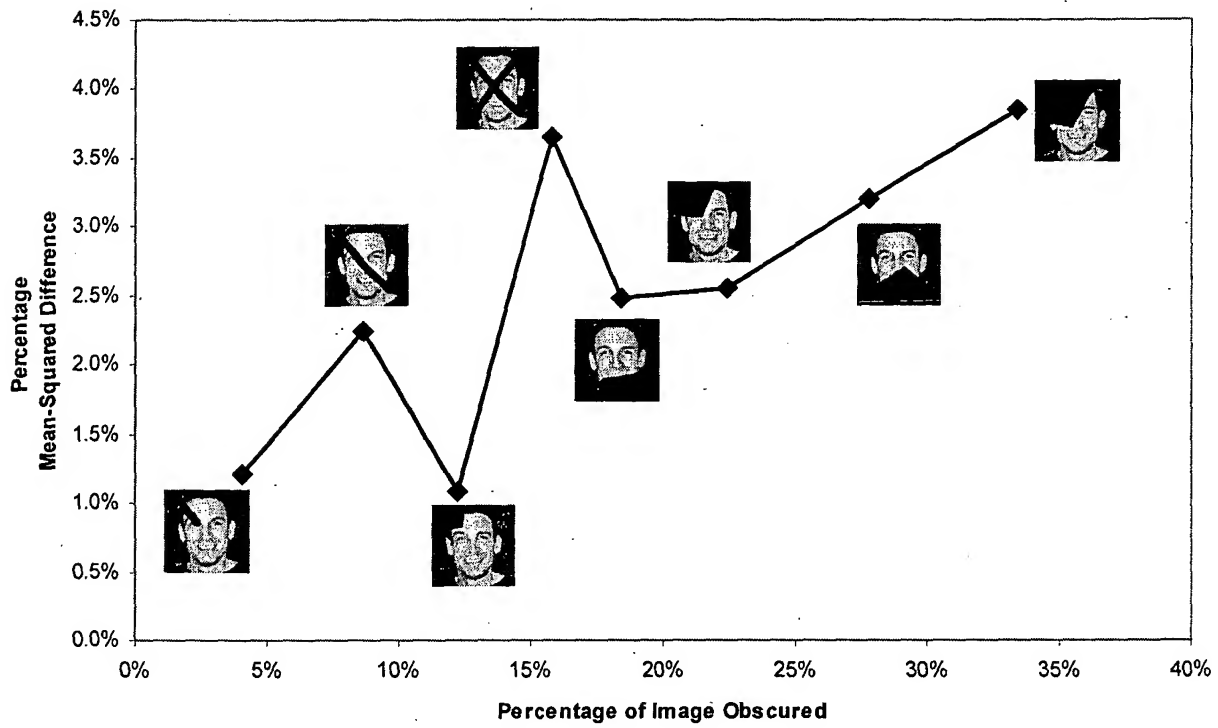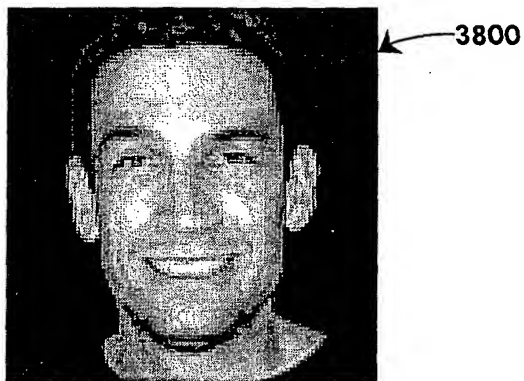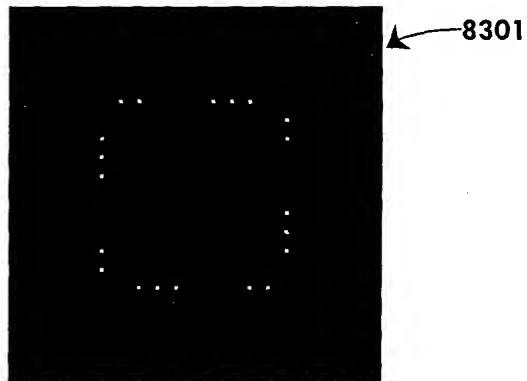
Figure 84a

5300

Figure 84b
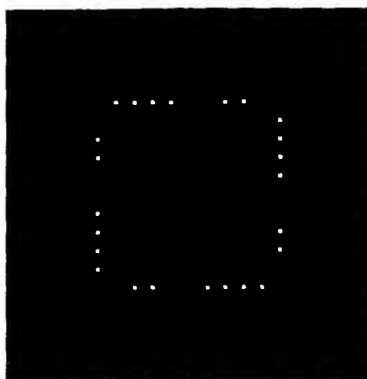
8401

Figure 85a

8500

Figure 85b
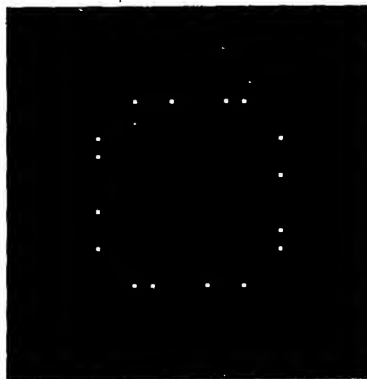
8501

Figure 86a

5400

Figure 86b
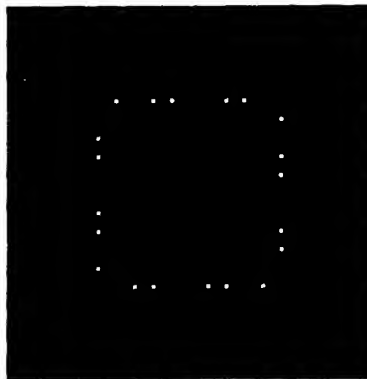
8601

Figure 87a

5500

Figure 87b
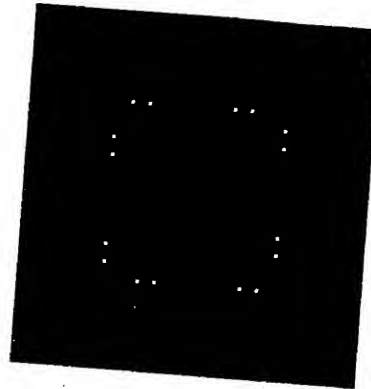
8701

Figure 88b
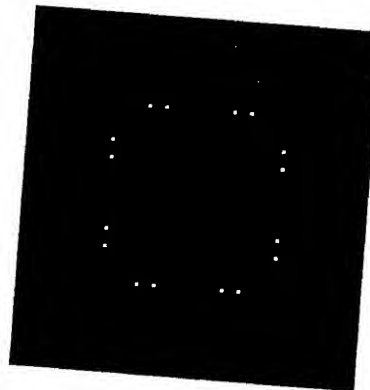
8800

Figure 88b

8801

Figure 89a

8900

Figure 89b

8901

Figure 90

9000

Figure 91

9100

Figure 92a

3800

Figure 92b

9201
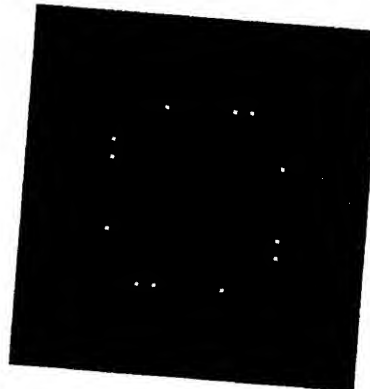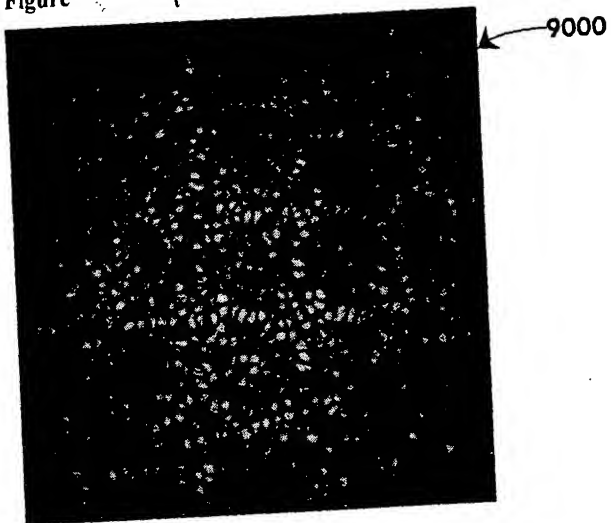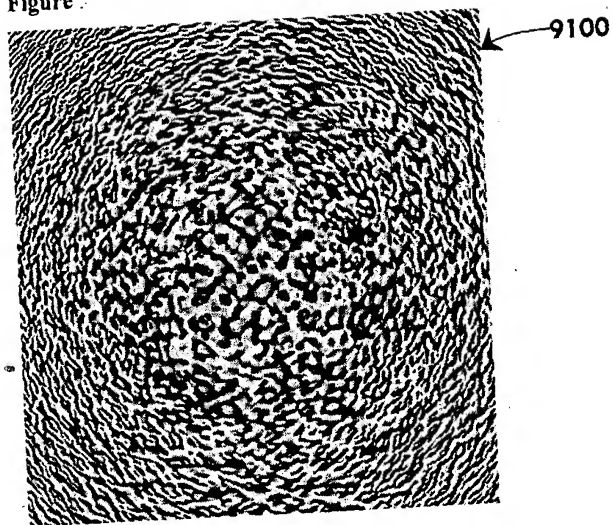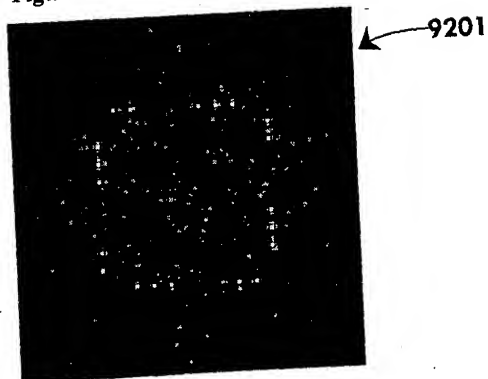
Figure 93a

5300

Figure 93b
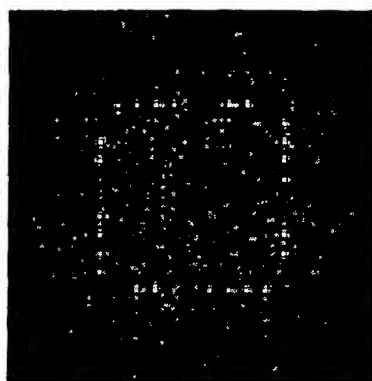
9301

Figure 94a

8500

Figure 94b
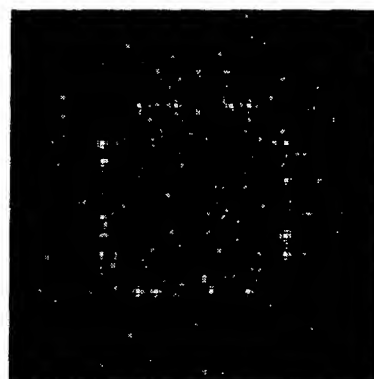
9401

Figure 95a

5400

Figure 95b
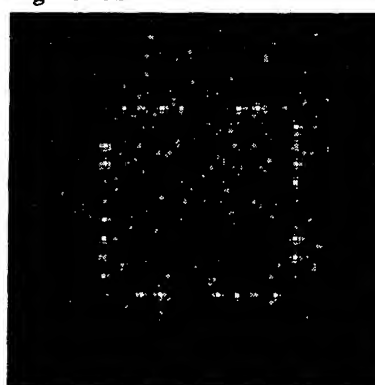
9501

Figure 96a

5500

Figure 96b
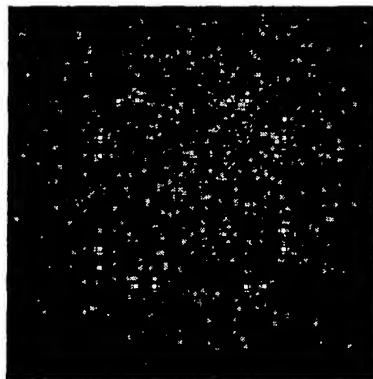
9601

Figure 97a

8800

Figure 97b
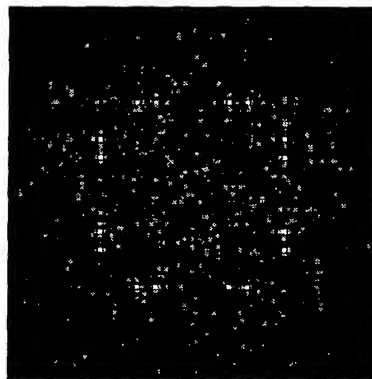
9701

Figure 98a

8900

Figure 98b
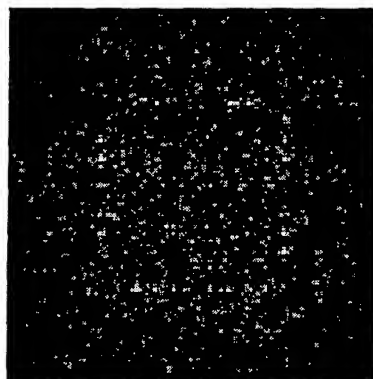
9801

Figure 99a

4800

Figure 99b
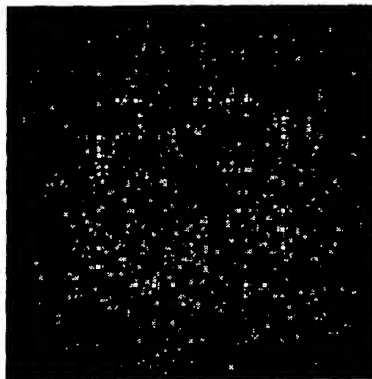
Figure 100a

4900

Figure 100b
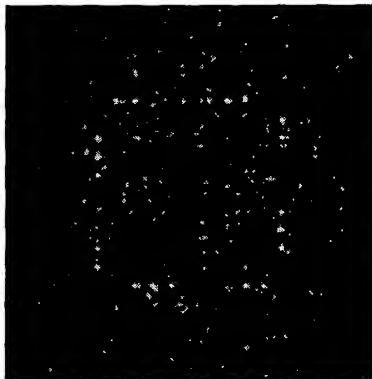
Figure 101a
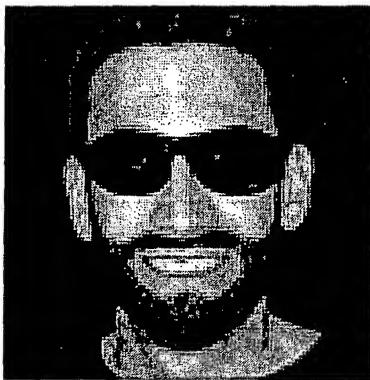
5000

Figure 101b
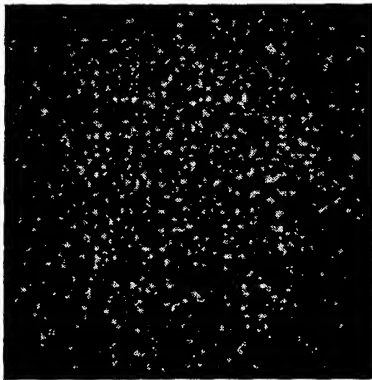
Figure 102a

5100

Figure 102b
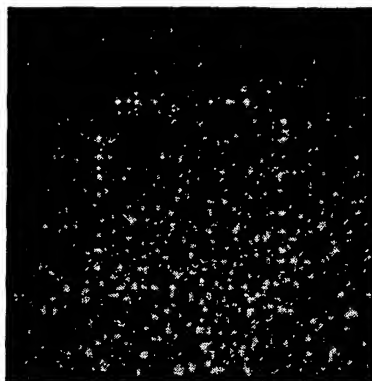
Figure 103a

5200

Figure 103b
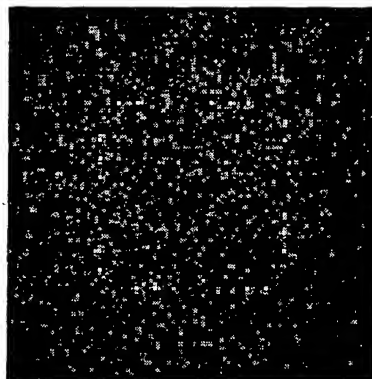
Figure 104a

10400

Figure 104b
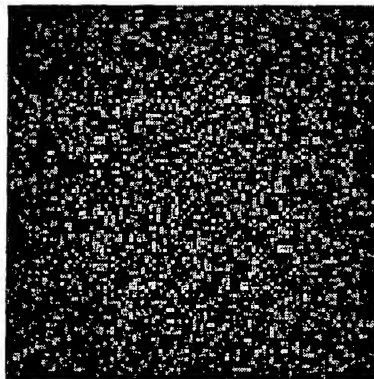
Figure 105

← 10500

Figure 106

← 10501

Figure 107

← 10502

Figure 108
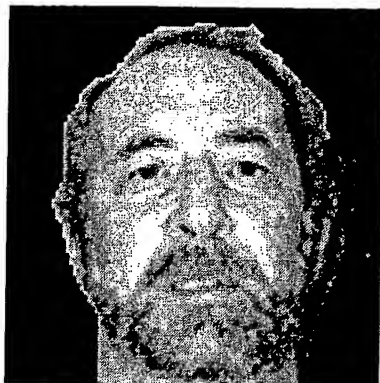
10503

Figure 107
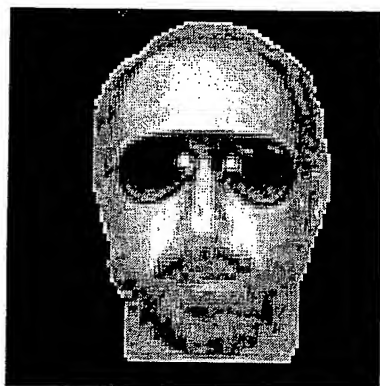
10504

Figure 110

11000

Figure 111

11100

Figure 112

11200

Figure 113

11300

Figure 114

←—11400

Figure 115
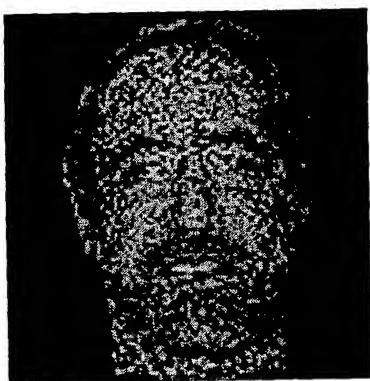
←—11500

Figure 116
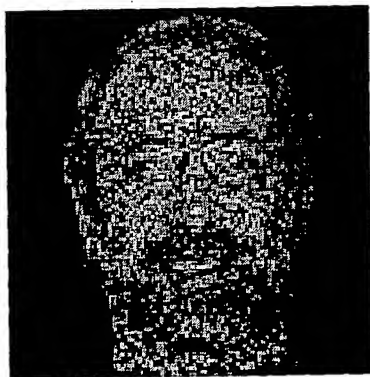
←—11600

Figure 117

11700

Figure 118a
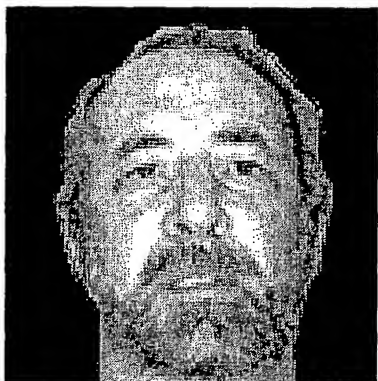
10503

Figure 118b
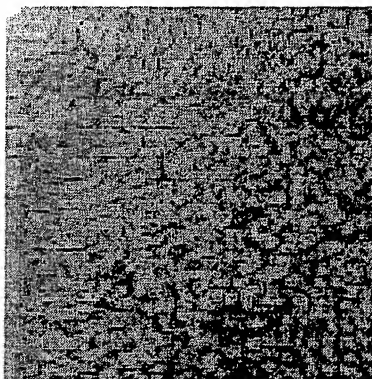
11801

Figure 119a

3800

Figure 119b
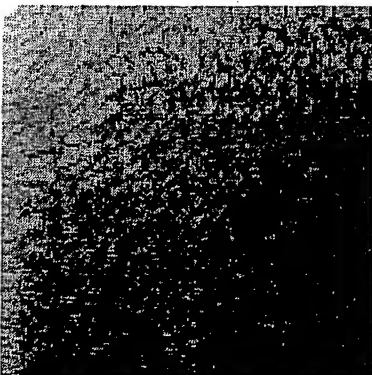
11901
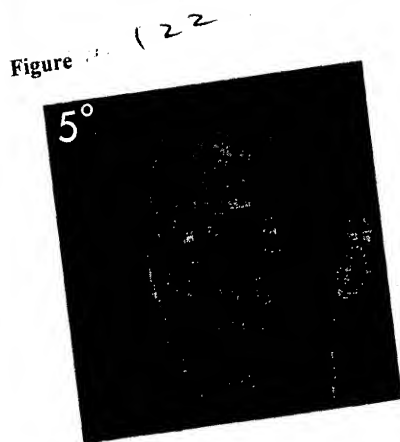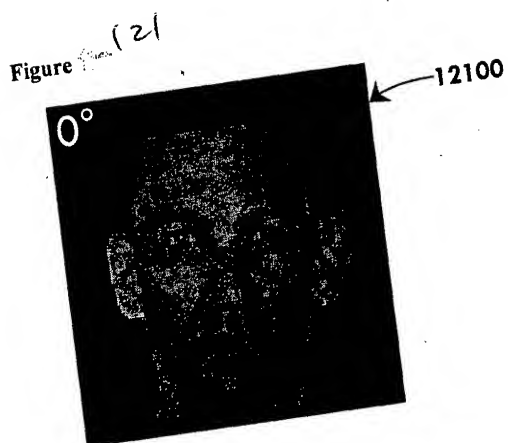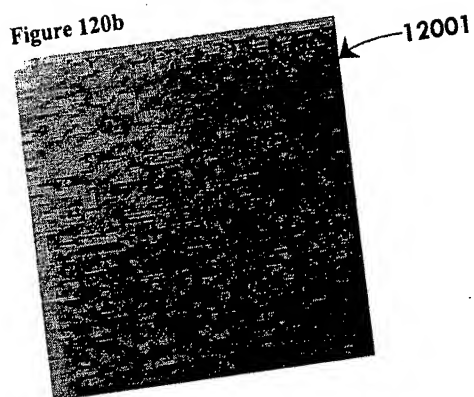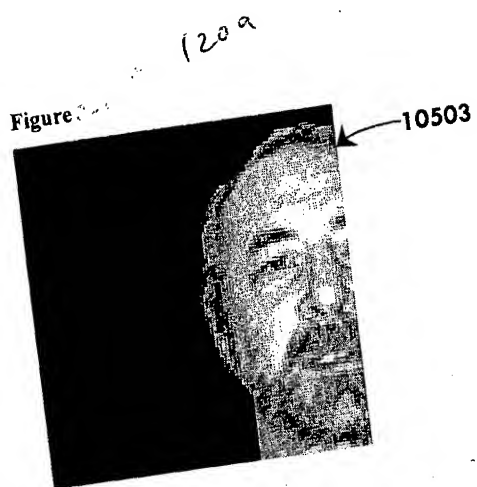
Figure 120a
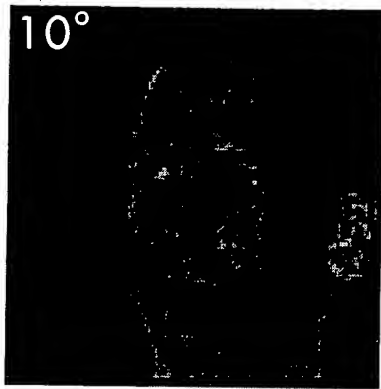
Figure: 〜 ←10503

Figure 120b ←12001

Figure: 121

Figure: 121 ←12100

0°

Figure: 122

Figure: 122

5°

Figure 123

10°

Figure 124

15°

Figure 125

20°

Figure (26

45°

Figure) (27

60° ←—12700

Figure 28

70°

Figure 129

80°
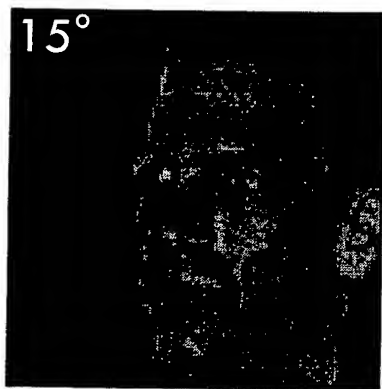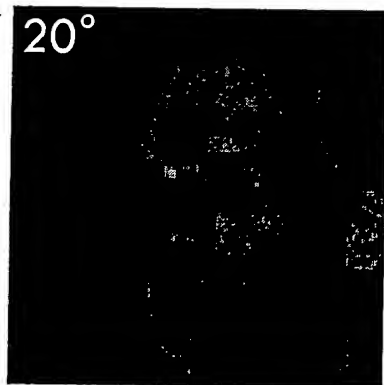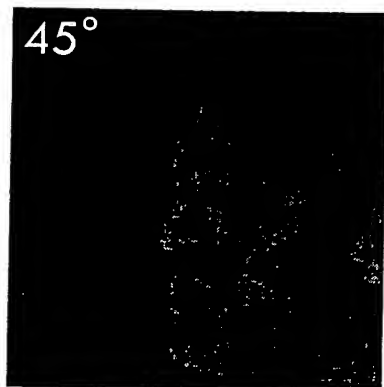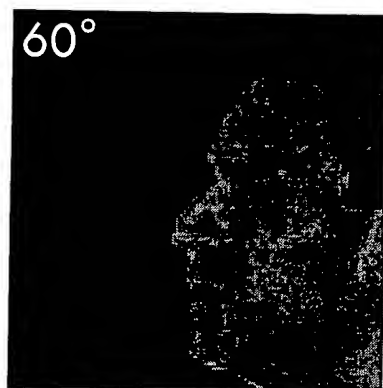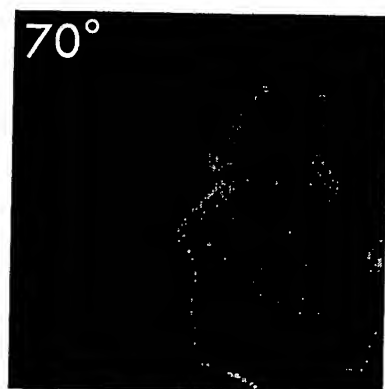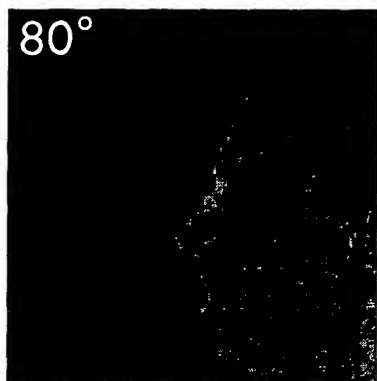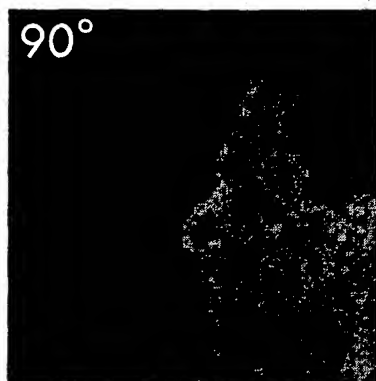
Figure 130

90°
← 13000

Figure 131

← 13100

Figure 132 132

132 133

Figure 133

134
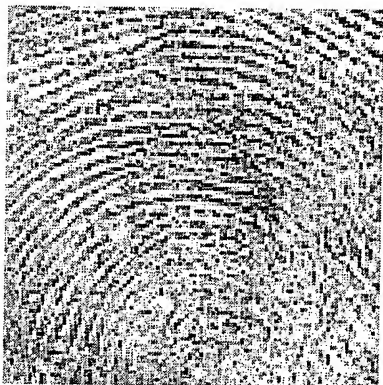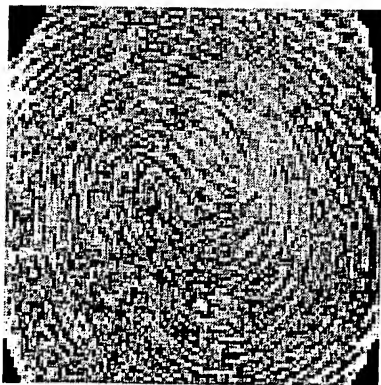
Figure

Figure 135

Figure 136

Figure 137
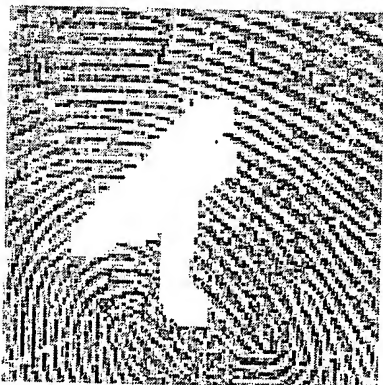
←—13700

Figure 138
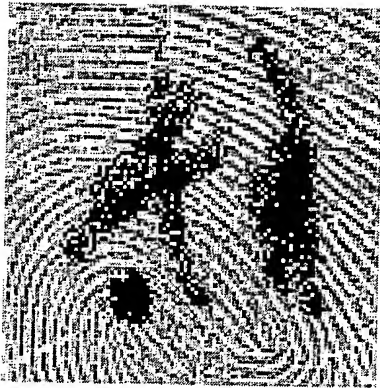
← 13800

Figure 139

← 13900

Figure 140

← 14000

Figure 141

14100

Stephen

Figure 142

Stephen

Figure 143

Stephen

Figure 144

Stephen 14400

Figure 145

Stephen

Figure 146

Stephen ←——14600

Figure 147

Stephen

Figure 148

Stephen

Figure 149

Stephen

Figure 150

Stephen

Figure 151

Paul ←—15100

Figure

152ª

Flowchart for software embodiment: Overall Control

Start

Initialize: Read disk file with parameters & transportation matrix into RAM ─15201

─15202
Permanent storage

─15203
A-tensor, λ, buffer and cortex dimensions, etc.

─15206
Defined, new, empty Cortex 103

─15205
Define C=0 (empty Cortex 103)

No

─15204
Use an existing cortex?

Yes

─15207
Read-in Cortex 103

─15202
Permanent storage

─15208
Retrieved Cortex 103 from disk

1
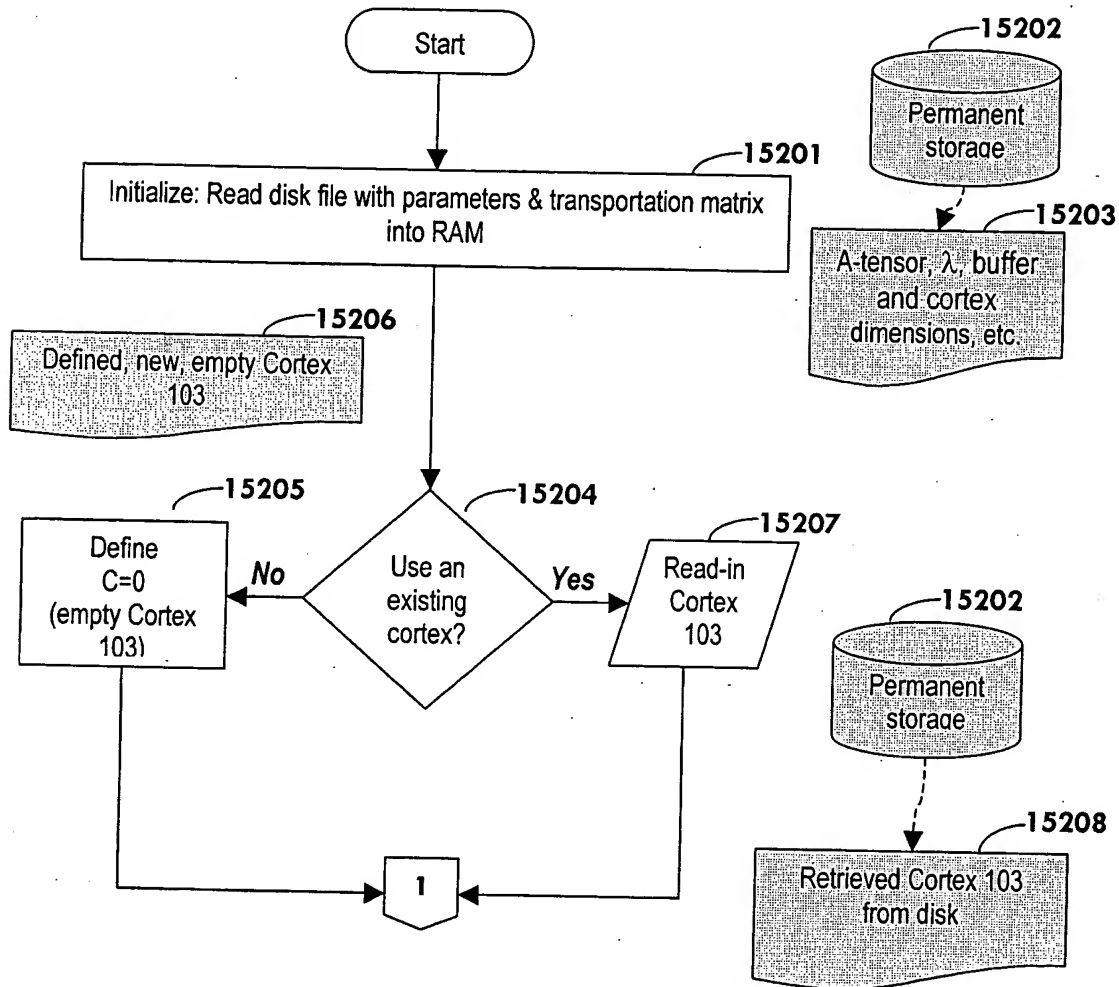
Flowchart for software embodiment: Overall Control, cont.

Figure 153

Flowchart for software embodiment: Create IIP

Subroutine **CreateIIP** ─15301

Get alphanumeric
label from keyboard

─15303

Convert label to 8-bit binary string
IB(i) of label's ASCII code

─15304

Call **DrawIIP** ─15306

─15305
IIP for the selected
ASCII code

Create new data file linked to ASCII
code for the IIP

─15308

─15307
New data file
opened

Get input to file
(photos, criminal
record, etc.) from
Keyboard

─15309
IIP and related
files

─15202
Permanent
storage

Store files and
IIP on disk ─15311

Return

Figure   1549

Flowchart for softwar   mbodiment: Learn

Subroutine **Learn**

—15401

Input file
specified from
keyboard

—15404

Call Import

—15407

Get IIP
alphanumeric label
from kevboard

—15408

Convert IIP label to 8-bit binary string
IB(i) of label's ASCII code

—15410

Call Import

2

—15202

Permanent
storage

—15403

Input file retrieved
from disk

—15405

Input file defined on
Buffer 102

—15406

Input file defined as
$C_1(i,j)$ for all points
(i,j) on Cortex 103

—15202

Permanent
storage

—15409

IIP file retrieved from
disk

—15411

IIP defined on IIP
Buffer 303

—15412

IIP defined as $C_2(i,j)$
for all points (i,j) on
Cortex 103

Flowchart for software embodiment: Learn (continued)

2

15413

Associate files:
$C_3(i,j)=C_1(i,j)*C_2(i,j)$ for all points (i,j)
on Cortex 103

15414

New association C_3
defined on Cortex
103

15415

Get fraction (α) of the new
association C_3 to add to the
total cortex file C on Cortex
103 from Keyboard

15416

Add association to total cortex array:
$C(i,j)=C(i,j)+\alpha*C_3(i,j)$ for all points
(i,j) on Cortex 103

15418

Contents of updated
cortex C defined on
Cortex 103

15417

Store updated total cortex file
C

15202

Permanent
storage

return

Flowchart for software embodiment: Recognize

Subroutine **Recognize**

— 15501

Input file to be Identified specified from keyboard

— 15202

Permanent storage

— 15503

Input file retrieved from disk

— 15504

Call **Import**

— 15505

Input file defined on Buffer 102

— 15507

Divide input array ϕ into full cortex array C for all points (i,j) on Cortex 103:

$$\psi_{ij} = \phi_{ij}^* C_{ij} \Big/ \left|\phi_{ij}\right|^2$$

— 15506

Input file defined as ϕ(i,j) for all points (i,j) on Cortex 103

— 15509

Call **Retrieve**

— 15508

Retrieved file defined as ψ(i,j) for all points (i,j) on Cortex 103

— 15511

Take magnitude of complex IIP array 15610

— 15510

IIP array defined on IIP Buffer 303

3

— 15512

Magnitude of IIP defined on IIP Buffer 303

Flowchart for software embodiment: Recognize (continued)

```
                    3
```

15513

| | Call ReadIIP | |

15514

Get file(s) from disk
using returned file
name 'IIPCode'

15515

File(s)
found?

No → **15516** Report failure to ID

Yes

15517 Display contents of file(s)

15202

Permanent
storage

15518

Identifying File(s)
Displayed

return

Figure   156a

Flowchart for software embodiment: Draw IIP

Subroutine **DrawIIP**

⌐15601

Define IIP filename: convert 8-bit
binary code IB(i) to a character string
and append ".raw"

⌐15603

IIP file for the
selected ASCII code

⌐15602

Create an IIP file with that name

⌐15604

Initialize IIP array:
BIIP(ix,iy)=0 for all points (ix,iy) on IIP
Buffer 303

4

Figure 156b

Flowchart for software embodiment: Draw IIP continued

```
                    ┌─────┐
                    │  4  │
                    └──┬──┘
                       │        ┌─15605
                       ▼
              ┌─────────────────┐
              │  ix=-25, iy=25  │
              └────────┬────────┘
                       │        ┌─15606
                       ▼
              ⬡─────────────────⬡
              │    Do j=1 to 8   │◄─────────────┐
              ⬡─────────────────⬡              │
                       │        ┌─15607         │
                       ▼                        │
                  ┌─────────┐                   │
                  │ ix=ix+5 │                   │
                  └────┬────┘                   │
                       │  ┌─15608    ┌─15609    │    ┌─15611
                       ▼                        │      Do i=1 to 4
                   ◇───────◇      ┌──────┐     ⬡───────────────⬡◄───────┐
                   │ ix=0? │─────►│ ix=5 │────►│   Do i=1 to 4  │        │
                   ◇───────◇ Yes  └───┬──┘     ⬡───────┬───────⬡        │
                       │ No           │                │   ┌─15612      │
                       ▼              │                ▼                 │
                      (○)◄────────────┘         ┌─────────────┐         │
                       │                        │   ix0=ix    │         │
                       │  ┌─15610               │   ix=iy     │         │
                       ▼                        │   iy=-ix0   │         │
                   ◇─────────◇                  │ BIIP(ix,iy)=255        │
                   │ IB(i)=1? │────────┐        └──────┬──────┘         │
                   ◇─────────◇  Yes    │               │                │
                       │ No            │              (○)◄──────────────┘
                       ▼               │               │
                      (○)◄─────────────┴───────────────┘
                       │
                       ▼
                 ╭──────────╮
                 │  Return  │
                 ╰──────────╯
```

┌─15613

Returned IIP buffer array BIIP(i,j)

Figure   157

Flowchart for software embodiment: Import

Subroutine **Import**

15701

Read specified disk file into
buffer array $B_0(i,j)$ for all points
(i,j) on Buffer 102

15202

Permanent
storage

15702

Input file retrieved
from disk

15703

Apply phase to create complex-valued
array: e.g.,
$B(i,j)=dcmplx(B_0(i,j)*cos(\beta*B_0(i,j)),$
$B_0(i,j)*sin(\beta*B_0(i,j)))$
for all points (i,j) on Buffer 102,
where β is an arbitrary phase
multiplier

15704

Complex array $B(i,j)$
defined on Buffer
102

15705

Loop over all
source points
(k,l) on Buffer
102

15706

Loop over all
destination
points (i,j) on
Cortex 103

15707

Calculate cortex array:
$C(i,j)= C(i,j) + A(i,j,k,l)*B(k,l)$
where is $A(i,j,k,l)$ is the transportation
tensor defined by Eq. 23

15708

Returned complex
array $C(i,j)$ defined
on Cortex 103

Return

Figure 158

Flowchart for software embodiment: Retrieve

Subroutine **Retrieve**

15801

Begin with array of complex numbers defined on Cortex 103

15802

Complex array C(i,j) defined on Cortex 103

15803

Loop over all source points (i,j) on Cortex 103

15804

Loop over all destination points (k,l) on Buffer 102

15805

Calculate buffer array:
B(k,l)= B(k,l) + A(k,l,i,j)*C(i,j)
where is A(i,j,k,l) is the transportation tensor defined by Eq. 23

Return

15806

Returned complex array B(i,j) defined on Buffer 102

Flowchart for software mbodiment: Read IIP

```
    ( Subroutine ReadIIP )
            |
            |  ⌐15901
            v
+-----------------------------+
| Take magnitudes in IIP Buffer 301 at  |
| the locations defined by the DrawIIP   |
| subroutine, according to Figure 157    |
+-----------------------------+
            |
            |  ⌐15903
            v
+-----------------------------+
| Find largest value (IIPMax(i)) and     |
| smallest value (IIPMin(i)) in each bar,|
| where i (1≤i≤4) indicates the iᵗʰ bar  |
+-----------------------------+
            |
            |  ⌐15904
            v
+-----------------------------+
|     Set initial breakpoint BP          |
|                                        |
|   BP=1/8 Σ⁴ᵢ₌₁ [ IIPMax(i) + IIPMin(i) ]|
+-----------------------------+
            |
            |  ⌐15905
            v
+-----------------------------+
|      Set iteration index n=0           |
+-----------------------------+
            |
            v
          [ 5 ]
            |
            |  ⌐15906
            v
+-----------------------------+
|          n = n + 1                     |
+-----------------------------+
            |
            |  ⌐15907
            v
+-----------------------------+
| Initialize confidence value v, which   |
| represents uncertainty in the          |
| identification, to 0                   |
+-----------------------------+
            |
            v
          [ 6 ]
```

15901

15903

15904

$$BP = 1/8 \sum_{i=1}^{4} [\ IIPMax(i) + IIPMin(i)\ ]$$

15905

15906

15907

⌐15902

Raw IIP values IIPRaw(i,j), where i (1≤i≤4) indicates the iᵗʰ bar and j (1≤j≤8) indicates the jᵗʰ bit

Flowchart for software embodiment: Read IIP (continued)

6

15908

Loop over IIP bits
Do j=1 to 8

15913

Tentative IIP bit values
IIPBit(i,j), where i (1≤i≤4)
indicates the ith bar and j
(1≤j≤8) indicates the jth bit

15909

Loop over IIP bars
Do i=1 to 4

15910

15911

IIPRaw(i,j) ≤ BP? —— Yes —— IIPBit(i,j) = 0

No

15912

IIPBit(i,j) = 1

7

Figure 159c

Flowchart for software embodiment: Read IIP (continued)

```
                    ┌───┐
                    │ 7 │
                    └─┬─┘
                      │                                    ┌─15913
                      ▼        ─15914          ┌─────────────────────────┐
              ╱──────────────╲                 │ Tentative IIP bit values│
             ╱ Loop over IIP   ╲               │ IIPBit(i,j), where i (1≤i≤4)│
             ╲ bits Do j=1 to 8 ╱              │ indicates the iᵗʰ bar and j │
              ╲──────────────╱   ─15915        │ (1≤j≤8) indicates the jᵗʰ bit│
              ╱──────────────╲                 └─────────────────────────┘
             ╱ Loop over IIP   ╲
             ╲ bars Do i=1 to 4 ╱
              ╲──────────────╱
                      │
                      ▼           ─15916                       ─15917
              ╱────────────────────╲        Yes          ┌──────────────┐
             ╱ 0.9*BP ≤ IIPRaw(i,j)  ╲───────────────────▶│  v = v + 1   │
             ╲  ≤ 1.1*BP?            ╱                     └──────────────┘
              ╲────────────────────╱
                      │ No
                      ▼
                    ( ○ )◀───────────────────────────────────
                      │
                      ▼          ─15918                        15919
              ╱────────────────────╲        Yes          ┌──────────────┐
             ╱ 1 of 4 bars disagree  ╲───────────────────▶│  v = v + 5   │
             ╲ for bit j (3:1 vote)? ╱                     └──────────────┘
              ╲────────────────────╱
                      │ No
                      ▼
                    ( ○ )◀───────────────────────────────────
                      │
                      ▼          ─15920                        15921
              ╱────────────────────╲        Yes          ┌──────────────┐
             ╱ 2 of 4 bars disagree  ╲───────────────────▶│  v = v + 100 │
             ╲ for bit j (2:2 vote)? ╱                     └──────────────┘
              ╲────────────────────╱
                      │ No
                      ▼
                    ( ○ )◀───────────────────────────────────
                      │
                      ▼
                    ┌───┐
                    │ 8 │
                    └───┘
```

Figure 159d

Flowchart for software embodiment: Read IIP (continued)

8

15923

$v \geq 100$ AND $n > 5$?

Yes → A

No

15922
Tentative IIP code IIPCode(j) based on current BP, where j ($1 \leq i \leq 8$) indicates the ith bit

15924
Report failure to make an identification

15925

$v \leq 4$ OR ($n > 5$ AND $v < 100$)?

Yes → B

No

15926
Report IIP code [IIPCode(j)] and confidence score (v)

15927
Call FindNewBreakPoint

Return

15928
Error returned?

Yes → A

No

15929
Has BP changed?

No →

15930
$v < 100$?

No → A

Yes → B

Yes

5

Figure 160

Flowchart for software embodiment: Find New BreakPoint

Subroutine **FindNewBreakPoint**

Discard all tentative IIP bit values
IIPBit(i,j) for which bit j has a 2:2 vote
— 16001

Rank all remaining bit values by size and
note whether each is designated a 0 or
1 (based on 3:1 or 4:0 vote)
— 16002

— 16003
Tentative IIP bit values
IIPBit(i,j), ranked by size

— 16304
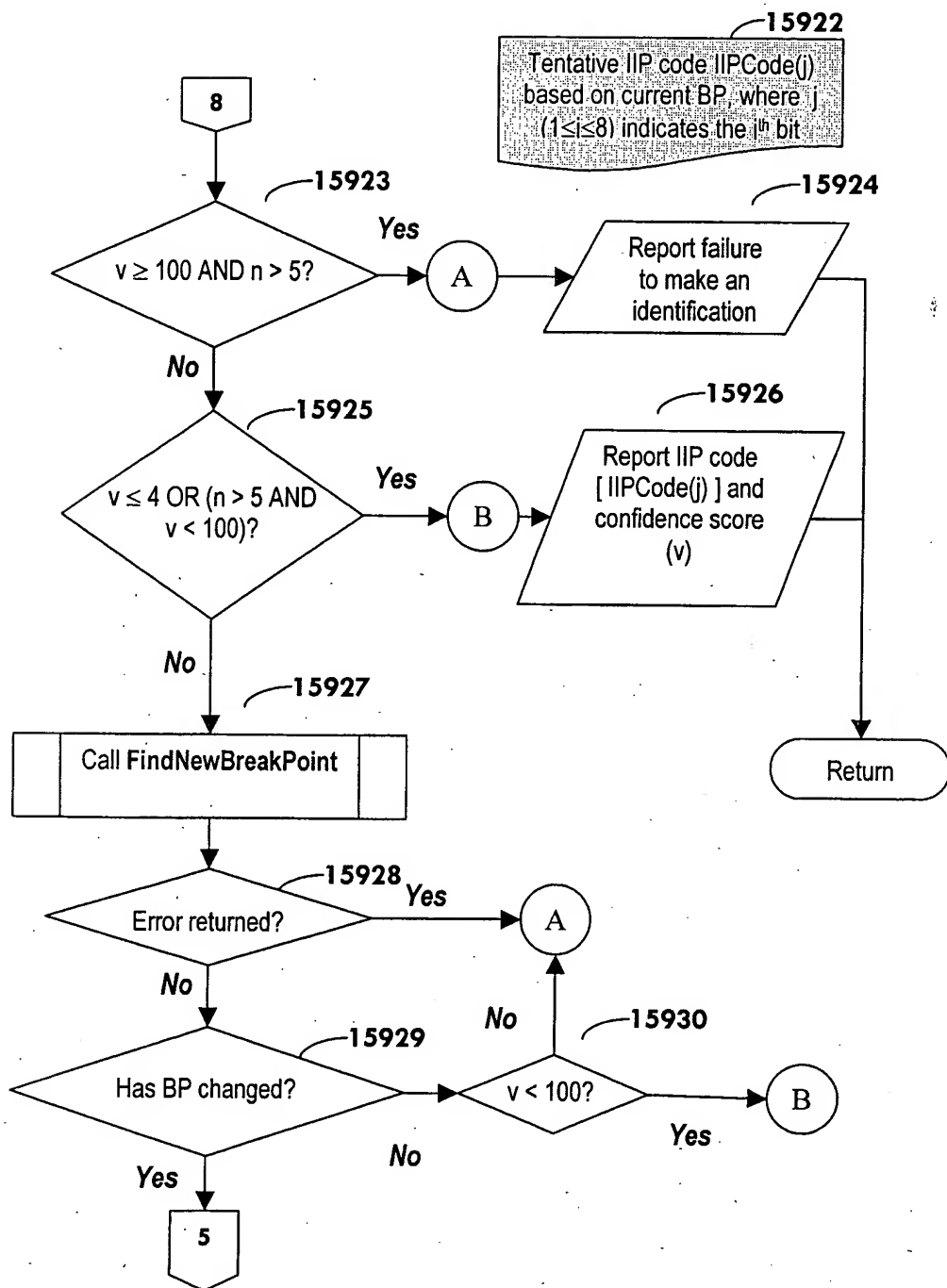Arrays of 0's
(IIPZeroes(k)) and 1's
(IIPOnes(k)), ranked by
size

— 16005
Normal distribution:
All 1's > all 0's?

Yes → — 16006
New BP=1/2 [Min(IIPOnes(k))
+ Max(IIPZeroes(k)))

No

— 16007
Slight overlap:
Largest 0 > smallest 1?

Yes → — 16008
New BP=1/2 (Min(IIPOnes(k))
+ Max(IIPZeroes(k))). Discard those two
values in IIPBit(i,j)

No

— 16009
Have three 1's or
three 0's already been
discarded?

Yes → — 16011
Report error → Return

No

— 16010
Discard abnormal value